Biblioteczka

Komputer Świat

Książka z płytą

NAUKA PROGRAMOWANIA PRZEZ ZABAWĘ

SCRATCH

NAUCZYSZ SIĘ:

■ układać własne programy ■ tworzyć zmienne, listy, pętle i instrukcje warunkowe ■ pisać własne gry ■ korzystać z Arduino ■ myśleć logicznie i kreatywnie

KŚ+

Z TĄ KSIĄŻKĄ – PŁYTA I E-WYDANIE GRATIS

Poniżej znajduje się płyta z kodem bonusowym. Wystarczy założyć konto i zalogować się na stronie ksplus.pl oraz zarejestrować kod, by uzyskać dostęp do e-wydania tej książki. Po zalogowaniu dodatkowo dostępne będą także do pobrania i zainstalowania wszystkie aplikacje opisane w książce.

Jeśli poniżej nie ma płyty z kodem, zwróć się do sprzedawcy

O braku płyty poinformuj również redakcję, pisząc na adres redakcja@komputerswiat.pl

Kod bonusowy należy zarejestrować w aplikacji KŚ+ (ksplus.pl)

Biblioteczka Komputer Świat

SCRATCH

NAUKA PROGRAMOWANIA
PRZEZ ZABAWĘ

KOMPUTER
ŚWIAT
BIBLIOTECZKA

od autora

Każdy powinien uczyć się programowania... ponieważ to uczy, w jaki sposób myśleć.
Steve Jobs

Piotr Szlagor

Scratch to środowisko do nauki programowania. Programując, rozwijamy wiele ważnych umiejętności, takich jak logiczne i strategiczne myślenie oraz umiejętność rozwiązywania problemów. To, co i ile stworzymy w Scratchu, zależy tylko od naszej wyobraźni.

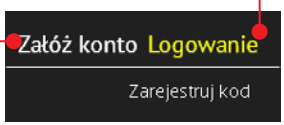
Ta książka ma na celu wprowadzenie do świata programowania w Scratchu. Zaczyna się od niezbędnych informacji, potrzebnych do rozpoczęcia pracy z tym środowiskiem programistycznym. Dzięki nim poznamy jego wszystkie niezbędne funkcje. Potem nauczymy się tworzyć kształty z wykorzystaniem funkcji rysowania w Scratchu. Tworząc rysunki, zobaczymy, jak działa pętla w programowaniu. Następnie przyjdzie czas na wspólne tworzenie bardziej skomplikowanych aplikacji. Zakodujemy kilka programów, a potem przejdziemy do tworzenia gier komputerowych. Po tym etapie tworzenie aplikacji nie będzie już miało przed nami żadnych tajemnic i będziemy mogli zająć się rozwiązywaniem zadań z programowania. Ostatni rozdział – to inspiracja na przyszłość. Znajdziemy w nim informacje, jak – znając już podstawy programowania – zacząć przygodę z tworzeniem urządzeń elektrycznych i aplikacji na smartfony.

Wersja online Scratcha jest dostępna za darmo w internecie, a wersję desktopową programu – do zainstalowania w komputerze, znajdziemy na płycie dołączonej do książki oraz w KŚ+. Na płycie i w KŚ+ znalazły się także projekty omówione w kolejnych rozdziałach i skrypty do wszystkich zadań.

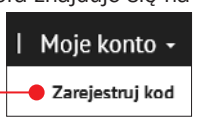
Miłej lektury!

OTO JAK SKORZYSTAĆ Z E-WYDANIA KSIĄŻKI

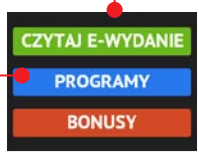
1 Otwieramy w przeglądarce stronę **www.ksplus.pl**. Logujemy się – możemy użyć konta z serwisu **Komputerswiat.pl**. Jeżeli nie mamy konta, klikamy na , aby się zarejestrować.



2 Po zalogowaniu się możemy zarejestrować unikalny kod nadrukowany na etykiecie płyty, która znajduje się na wewnętrznej stronie okładki. Wystarczy kliknąć na link i przepisać kod.



3 Możemy korzystać z e-wydania i mamy pełny dostęp do programów opisanych we wskazówkach. Do KŚ+ możemy logować się zawsze i wszędzie.



SCRATCH
NAUKA PROGRAMOWANIA PRZEZ ZABAWĘ

ROZDZIAŁ 1

Dlaczego warto poznać Scratcha

Według twórcy Scratcha, Mitchela Resnicka, nauka programowania jest dziś tak ważna jak nauka pisania. Dlaczego warto uczyć się tworzyć programy?

Programista to dziś jeden z najbardziej poszukiwanych zawodów. Warto się więc uczyć programowania już od najmłodszych lat. Ta umiejętność przyda się nawet, jeśli wybierzemy inny kierunek rozwoju. Tworzenie programów jest rozwijającym, kreatywnym zajęciem. Najprostszą drogą, prowadzącą do szybkiego nauczenia się budowania aplikacji, jest rozpoczęcie przygody ze Scratchem – prostym i darmowym środowiskiem, stworzonym z myślą o edukacji. Skrypty można w nim układać jak z klocków. Przekonajmy się, jakie to przyjemne zajęcie.

DROGOWSKAZ

Programowanie – do czego mi się to przyda

Około 100 poleceń, które da się ogarnąć wizualnie, sprawia, że możesz stworzyć wszystko, co chcesz, i nauczyć się podstaw zaawansowanych języków programistycznych. Dodatkowo istnieje cała społeczność ludzi gotowych pomóc ci w projektach i chętnie odpowiadających na pytania.

14-latek

Nauka programowania stała się na przestrzeni kilku ostatnich lat bardzo popularnym zajęciem. Co rusz powstają szkoły programowania dla najmłodszych, uczniów szkół średnich czy nawet kursy dla osób dorosłych. Co takiego szczególnego jest w tej umiejętności? Dlaczego powstają takie programy, jak Mistrzowie Kodowania, uczące programowania dzieci w każdym wieku, i skąd się bierze ich popularność? Odpowiedzi na te pytania można mnożyć.

Przede wszystkim programowanie zwyczajnie uczy i rozwija. Tworzenie własnych aplikacji doskonale kształtuje umiejętność myślenia strategicznego, rozwiązywania problemów i kreatywnego podejścia do stawianych zadań. Zauważmy, że te kompetencje mają znaczenie nie tylko w wypadku informatyki. Są po prostu potrzebne w życiu, i to na każdym jego etapie – czy w szkole, w pracy czy nawet podczas zakupów w sklepie. Świetnie ujął to Bill Gates: *Nauka pisania programów to gimnastyka dla mózgu. Pozwala wypracować umiejętność efektywnego myślenia o rzeczach niezwiązanych z informatyką.*

Praca nad tworzeniem aplikacji uczy też przydatnych w życiu kompetencji międzyludzkich. Pracując wspólnie nad zadanym projektem, ćwiczymy umiejętność współpracy, komunikacji i bardzo często cierpliwość. Korzyścią z nauki programowania jest też większa aktywność oraz motywacja do rozwiązywania zadań. Można to uzasadnić tym, że w wypadku programowania dosta-

Serwis Code Studio – samouczek dla początkujących informatyków

jemy efekt swoich prac od razu – program albo spełnia nasze oczekiwania, albo należy pewne jego elementy poprawić, by móc cieszyć się ze swojego sukcesu.

Nauka programowania opłaca się również ze względów finansowych. W 2013 roku aż cztery z dziesięciu najbardziej poszukiwanych zawodów wiązały się z umiejętnościami programistycznymi. Duża część przedsiębiorstw zakłada, że ich pracownicy będą musieli pracować z wykorzystaniem komputera – mało tego, coraz częściej wymaga się od pracownika znajomości i umiejętności obsługi specjalistycznego oprogramowania. Jak się okazuje – firmy te mają problemy ze znalezieniem odpowiedniej kadry.

Szybkie wejście w świat programowania

Ja i mój ośmioletni brat jesteśmy aktywnymi członkami społeczności Scratcha już od ponad roku. Bardzo lubię projektować gry i animacje z innymi dziećmi, bo różne osoby mają różne talenty i kiedy pracujemy razem, możemy stworzyć lepsze programy, niż gdybyśmy pracowali sami. **13-latek**

Scratch jest narzędziem programistycznym bardzo prostym i intuicyjnym w obsłudze. Łatwością, z jaką zaczyna się tworzyć w nim aplikacje, zadziwia każdego. Jak łatwo będzie można niedługo zauważyć, możliwości Scratcha są ograniczone głównie przez naszą wyobraźnię oraz umiejętność przełożenia jej na odpowiednie ułożenie klocków. Tę umiejętność da się bardzo łatwo wyćwiczyć. Wystarczy tworzyć coraz to nowe programy. Dobrym narzędziem, aby poćwiczyć umiejętności programistyczne, jest serwis **Code Studio** (**codestudio.org**). To gigantyczny samouczek dla początkujących informatyków, stworzony przez inżynierów z Google, Facebooka, Microsoftu i Twittera. Jest tu mnóstwo różnego rodzaju ćwiczeń, polegających na ułożeniu bloczków funkcjonalnych tak, by spełniały założenia podanego zadania. Mało

Scratch na iPady oraz na tablety z systemem Android, czyli ScratchJr

tego – w zadaniach występują postacie znane z gier i bajek, co stanowi dodatkową atrakcję dla młodszych odbiorców.
Koniecznie wspomnieć należy również o specjalnie przygotowanej wersji Scratcha na iPady i tablety z systemem Android. ScratchJr, bo o nim mowa, to nowa aplikacja przeznaczona dla dzieci w wieku 5–7 lat. To uproszczona wersja „dorosłego" Scratcha, w której da się tworzyć interaktywne historie i gry. Nie ma chyba lepszego sposobu na wyrażenie swojej osobowości w tak młodym wieku.

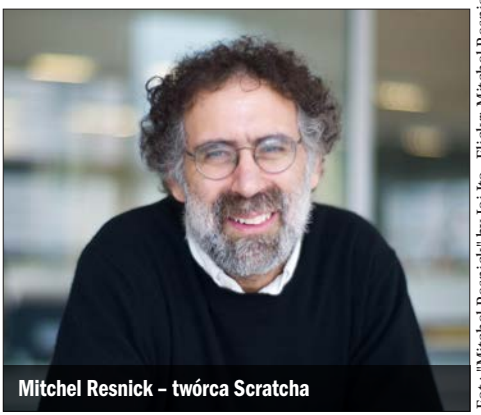

Lekcja ze Scratchem w szkole podstawowej

Dlaczego Scratch?

Wolę Scratcha bardziej niż blogi czy serwisy społecznościowe takie jak Facebook, bo tworzymy w nim interesujące gry i projekty, które są fajne do zabawy, oglądania i pobrania. Nie lubię rozmawiać z innymi o niczym. Wolę rozmawiać o czymś kreatywnym i nowym. **13-latek**

Scratch to środowisko programistyczne i jednocześnie społeczność internetowa, gdzie dzieci mogą programować i udostępniać swoje interaktywne materiały, takie jak historyjki, gry i animacje, ludziom z całego świata. Ujrzał on po raz pierwszy światło dzienne w 2006 roku, opublikowany przez zespół Lifelong Kindergarten (trudno przetłumaczyć tę nazwę na język polski – aby oddać jej sens, trzeba by ją przełożyć jako Całożyciowe przedszkole) pod kierownictwem Mitchela Resnicka.

Jego twórcom przyświecało hasło: Powinniśmy sprawić, by nasze szkoły, a nawet całe nasze życie bardziej przypominało przedszkole. I trzeba przyznać, że udało im się to założenie zrealizować.
W założeniach Scratch był przeznaczony dla dzieci w wieku od 8 do 16 lat. Jak pokazuje jednak praktyka – spokojnie mogą używać go dzieci w wieku przedszkolnym, jak również licealiści i studenci. Scratch pozwala łatwo tworzyć aplikacje mogące wykorzystywać dźwięki, rysować, porównywać ko-


Mitchel Resnick – twórca Scratcha

Fot.: "Mitchel Resnick" by Joi Ito - Flickr: Mitchel Resnick/Wikimedia Commons

Strona internetowa Scratcha

lory i wiele więcej. Jest całkowicie bezpłatny i trudno by było znaleźć komputer, na którym by nie działał. Czyni go to dostępnym dla każdego. Wymaganiem opcjonalnym, pozwalającym Scratchowi rozwinąć swój pełen potencjał, jest dostęp do internetu. Wtedy można swoje aplikacje udostępniać internautom na całym świecie, przeglądać programy innych użytkowników, dyskutować o nich czy remiksować je.

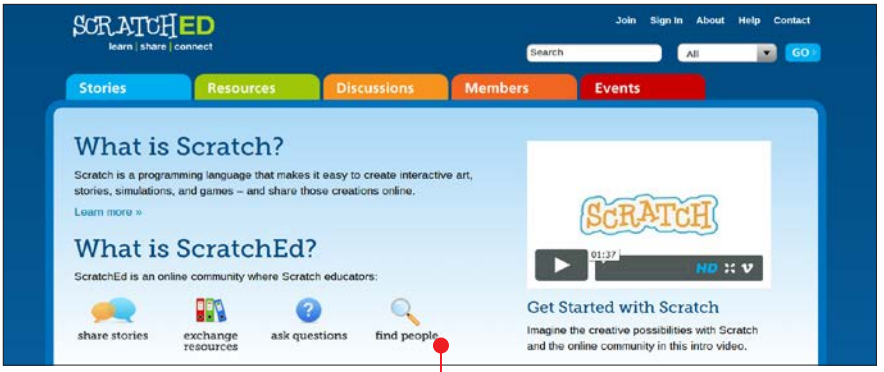
Programowanie w Scratchu polega na budowaniu list instrukcji z bloczków. Jest to chyba najwygodniejszy i najbardziej przemyślany sposób na tworzenie skryptów dla osób początkujących, jaki może być. Nie wymaga on bowiem od użytkownika żadnej wcześniejszej znajomości języków programowania. Wszystkie komendy są wypisane na bloczkach, w języku polskim. Mają one, w zależności od tego, do czego są przeznaczone, różne kolory. Dzięki temu struktura każdej tworzonej aplikacji znacznie zyskuje na przejrzystości. Tak wyglądający kod będzie z pewnością wspomagał naukę programowania.

Scratch jest bardzo popularny w naszym kraju. Duży wpływ miały na to liczne programy promujące programowanie wśród uczniów szkół. W tej książce znajdziemy materiały dla początkujących i przykłady ćwiczeń.

Scratch 2.0 – co nowego?

Scratch już dawno zaskarbił sobie sympatię na całym świecie. W swojej wersji 2.0 to masa nowych funkcji w porównaniu z wersją 1.4. Jest ich zbyt wiele, by zmieścić je w krótkim zestawieniu, lecz warto wymienić tu kilka najważniejszych. Oto one:

- Praca w nowym Scratchu może opierać się teraz wyłącznie na projektowaniu w internecie. Scratch jest już w pełni aplikacją chmurową.
- Od teraz można tworzyć własne bloczki, gdy tylko zajdzie taka potrzeba. Znacznie przyspiesza to proces projektowania programów i stwarza nowe możliwości dla twórców aplikacji.
- Jeśli spodoba się nam program innej osoby, możemy podejrzeć sposób, w jaki został on wykonany. Da się to zrobić jednym kliknięciem na stronie projektu tego użytkownika.
- Wprowadzono zmienne globalne, pozwalające na korzystanie z bazy danych. Dzięki temu rozwiązaniu da się tworzyć programy obsługujące na przykład listę najwyższych wyników w danej grze, widoczną dla użytkowników na całym świecie.
- Grafiki używane w projektach to już grafiki wektorowe – oprawa wizualna jest



teraz ładniejsza, a pliki zajmują mniej miejsca i ładują się szybciej.

Od teraz wszystkie poprawki i udoskonalenia do Scratcha będą wprowadzane w sposób niezauważalny i niewymagający od użytkownika żadnych działań. Gdy otworzymy stronę Scratcha, zawsze uruchomi się najnowsza wersja oprogramowania. Od niedawna jest też możliwa do zainstalowania wersja offline. To ukłon w stronę użytkowników komputerów bez stałego dostępu do internetu.

Gdzie szukać pomocy?

Początki pracy w Scratchu nie powinny nikomu przysporzyć kłopotów. Korzystając z tej książki i strony WWW Scratcha, można szybko stać się ekspertem i samemu zacząć pomagać innym. Warto jednak poznać kilka serwisów, gdzie spotkamy życzliwe osoby, które chętnie służą radą, czy zwyczajnie poczytać, jak inni pracują z tym środowiskiem. Koniecznie należy tu wspomnieć o siostrzanym serwisie portalu Scratcha – **ScratchEd** (**scratched.gse.harvard.edu**). Jest to duża sieć osób pracujących z tym środowiskiem programistycznym. Dewizą strony jest hasło Ucz, dziel i łącz się. Znaleźć można tam ludzi dzielących się swoimi doświadczeniami, projektami czy materiałami, których używają na swoich zajęciach. W ScratchEd, który działa od 2009 roku, został opublikowany już ogrom przydatnych materiałów. Warto poświęcić chwilę, by przejrzeć ten serwis. Z pewnością pomoże nam on w pracy ze Scratchem.

Szukając dobrych materiałów edukacyjnych, trzeba też koniecznie zajrzeć na stronę projektu **Mistrzowie Kodowania** (**mistrzowie-kodowania.pl**). Jest to serwis ogólnopolskiego programu popularyzującego naukę programowania w szkołach już od najmłodszych lat. W projekcie z roku na rok bierze udział coraz więcej szkół. Przeglądając portal, można znaleźć dużo scenariuszy zajęć przygotowanych dla różnych grup wiekowych – od pierwszej klasy szkoły podstawowej do końcowych klas gimnazjum. Wszystkie materiały tu umieszczone są na licencji Creative Commons, co oznacza, że można je spokojnie przygarnąć i z nich korzystać.

ROZDZIAŁ 2

Pierwsze kroki ze Scratchem

Scratch jest bardzo przyjaznym środowiskiem programistycznym, ale początki zawsze mogą być nieco trudne. Przeczytajmy, jak zacząć przygodę ze Scratchem

Ten rozdział zawiera niezbędne podstawy – ABC obsługi Scratcha i tworzenia w nim programów. Poznamy takie pojęcia, jak zmienne, listy czy pętle. Nauczymy się publikować interaktywne aplikacje w internecie i tworzyć własne bloczki funkcyjne. Dowiemy się wszystkiego, co potrzebne, by móc postawić pierwsze kroki na ścieżce informatycznej kariery. **Uwaga! Ze Scratcha można korzystać online, jak opisano to na stronie obok. Możemy również zainstalować Scratcha z płyty dołączonej do książki, pobrać z KS+** lub ze strony domowej projektu (patrz strona 13).

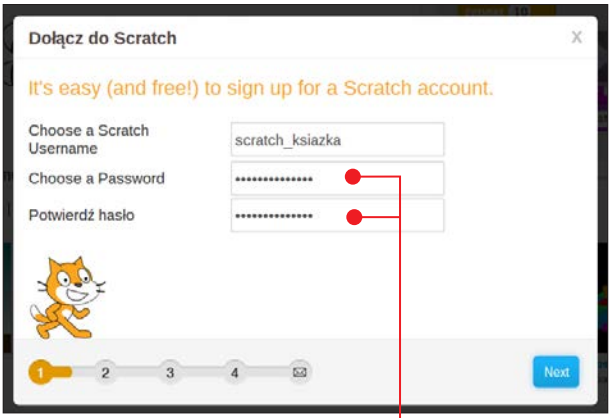
DROGOWSKAZ

Tworzenie i uzupełnianie konta w serwisie Scratch

Pracę ze Scratchem zaczniemy od stworzenia konta w serwisie tego projektu. Pozwoli to na tworzenie swoich programów i ich automatyczne zapisywanie w chmurze oraz na pracę zawsze w najnowszej wersji aplikacji. Będziemy mogli dodatkowo uruchamiać, komentować i remiksować projekty innych użytkowników.

Scratch jest dostępny dla każdego, kto ma komputer z dostępem do internetu oraz przeglądarkę internetową. Nie trzeba się martwić systemem operacyjnym, nie trzeba też przeznaczać miejsca na komputerze na jakiekolwiek aplikacje. Wszystko jest dostępne od ręki i trzeba przyznać – jest to naprawdę bardzo wygodne.

Aby założyć swoje konto w serwisie Scratch, należy wykonać następujące czynności:



1 Włączamy stronę projektu Scratch: **scratch.mit.edu**

2 Klikamy na przycisk **Dołącz do Scratch**.

3 W oknie, które się pojawi, wpisujemy nazwę użytkownika, a następnie dwukrotnie hasło.

4 W kolejnym kroku podajemy datę urodzenia, płeć oraz kraj, z którego pochodzimy.

UWAGA!

Nazwa użytkownika musi być niepowtarzalna w zakresie całego serwisu – prawdopodobnie próba użycia nazwy Ania zakończy się niepowodzeniem, w takim wypadku dodajmy do nazwy jakieś znaki, pamiętając o tym, że możemy wykorzystywać tylko litery, cyfry i podkreślnik

5 W kroku trzecim musimy wpisać swój aktualny adres e-mail. Nie trzeba obawiać się żadnego spamu. E-mail podajemy wyłącznie w celach weryfikacyjnych oraz na wypadek, gdybyśmy kiedyś zapomnieli hasła do naszego konta.

6 Sprawdzamy teraz skrzynkę pocztową. Został na nią wysłany e-mail z linkiem potwierdzającym. Klikamy na podany link, by potwierdzić założenie

Nowy profil użytkownika w serwisie Scratch

konta i uzyskać dostęp do pełnej funkcjonalności serwisu Scratch.

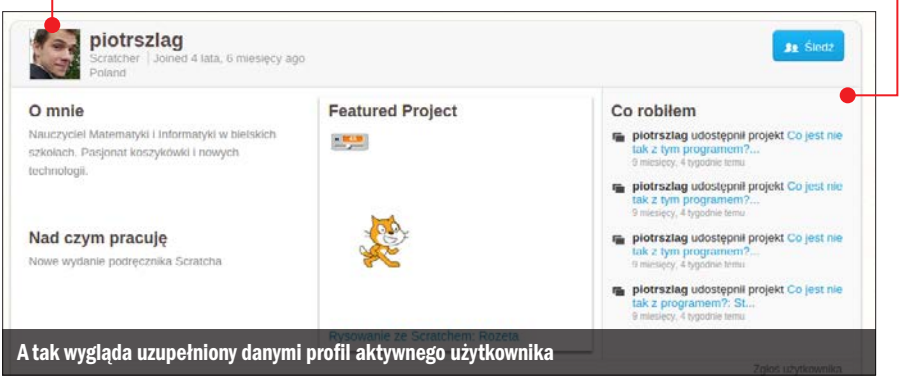
Personalizujemy konto

Kiedy mamy już konto, możemy je spersonalizować.

1 Klikając w prawym górnym rogu okna serwisu na naszą nazwę użytkownika, a następnie na **Profil**, otworzymy naszą wizytówkę, którą będą oglądali inni użytkownicy. Możemy ją edytować:

- Zmienić awatara. Robimy to, klikając na ikonę kota ●, a następnie wybierając dowolny plik graficzny ●.
- Podać trochę informacji o sobie – na przykład skąd jesteśmy czy też jakie są nasze zainteresowania.
- Pochwalić się swoimi planami związanymi z programowaniem w Scratchu, choćby wpisując w pozycji **Nad czym pracuję**, na przykład – Najlepsza gra komputerowa na świecie.

2 Przykładowy, uzupełniony profil może wyglądać tak jak na ilustracji poniżej ●. Jak widać, jest tu również lista naszych najnowszych projektów oraz informacje o działaniach podejmowanych w serwisie.



A tak wygląda uzupełniony danymi profil aktywnego użytkownika

Scratch bez internetu – instalacja edytora offline

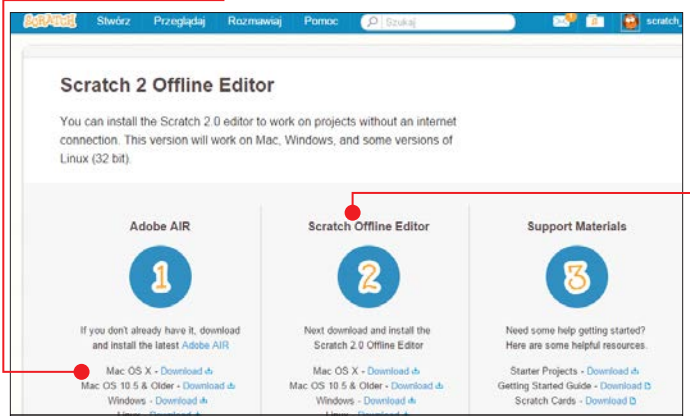
Komputery bez stałego dostępu do internetu można z powodzeniem wyposażyć w offline'owy edytor Scratcha. Niedawno została opublikowana jego najnowsza wersja. Instalacja jest dziecinnie prosta i bardzo szybka. Edytor działa na systemach Windows, Linux i Mac OS X. Sam się aktualizuje (po uzyskaniu dostępu do internetu i pozwolenia od użytkownika), więc nie trzeba odwiedzać strony Scratcha i sprawdzać, czy twórcy dodali nowszą wersję programu.

1 Przede wszystkim **możemy Scratcha zainstalować z płyty dołączonej do książki** (więcej o płycie na stronie 29). Możemy także pobrać go i zainstalować z KŚ+. W tym celu musimy zarejestrować na stronie **www.ksplus.pl** kod nadrukowany na płycie dodanej do książki. Informację, jak to zrobić, znajdziemy na samym początku książki – na stronie 2. Trzeci sposób – to pobranie programu ze strony internetowej o adresie **scratch.mit.edu/scratch2download**.

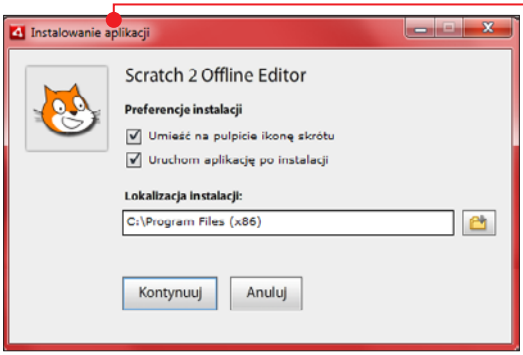
2 Scratch wymaga zainstalowanego na komputerze środowiska Adobe Air. Czy je mamy, możemy sprawdzić, otwierając listę zainstalowanych programów **Programy i funkcje** w Panelu sterowania. Jeżeli mamy Adobe Air, możemy przejść do instalacji Scratcha. Jeśli nie mamy, możemy go zainstalować z płyty, z KŚ+ lub ze strony internetowej Scratcha, wybierając wersję odpowiednią dla naszego systemu.

3 Jeżeli chcemy pobrać Scratcha z jego strony, robimy to, klikając na pole oznaczone cyfrą 2. Wybieramy wersję odpowiednią dla naszego systemu operacyjnego, pobieramy plik instalacyjny (około 40 MB) i uruchamiamy go.

4 Otwiera się okno kreatora instalacji. Decydujemy, czy program ma dodać na pulpit ikonę Scratcha. Klikamy na **Kontynuuj**.



Strona domowa offline'owego edytora Scratcha

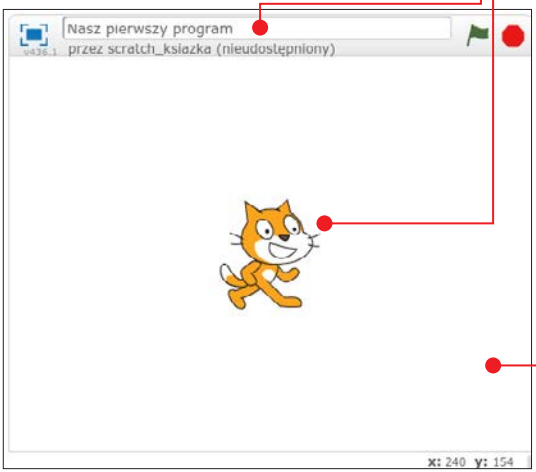


5 Zaczyna się instalacja. Po jej ukończeniu program jest gotowy do pracy.

6 Jeśli program otworzy się z napisami w języku angielskim, możemy łatwo przestawić go na język polski. W lewym górnym rogu okna aplikacji klikamy na ikonę ustawień i wybieramy język polski.

Tworzymy pierwszy program

Mamy już Scratcha – teraz zajmijmy się stworzeniem pierwszego działającego programu. Podczas tworzenia nowych programów w Scratchu zawsze w centrum ekranu będzie pojawiał się pomarańczowy kot. To właśnie on będzie wykonywał zadania, których instrukcje mu ułożymy. Wszystkie postaci nazywają się w Scratchu **duszkami**. Będziemy animować postać kota – niech zrobi kilka kroków i zagra na bębenku. Animacja nie będzie może bardzo efektowna, ale na razie mamy inny cel. Jest nim poznanie samego procesu układania skryptów, ich uruchamiania oraz ulepszania. A zatem – do dzieła.



Poznajemy okno Scratcha

1 Logujemy się na stronie **scratch.mit.edu** na swoje konto i klikamy na **Stwórz** w górnej części okna serwisu. Otwiera się **okno tworzenia programów**. Możemy również uruchomić edytor offline i tak rozpocząć pracę z aplikacją. (Całe okno Scratcha zobaczymy na stronach 18–19).

2 Poznajmy elementy tego okna. Oto panel wykonywania programu. To tu będą wizualizowane ułożone przez nas instrukcje. Dokładnie taki sam ekran zobaczą też użytkownicy z całego świata, uruchamiając naszą aplikację. Warto więc dbać

o jego estetykę. U góry widać pole z nazwą naszego programu. Nie zapominajmy go uzupełnić.

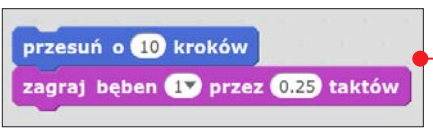
Powyżej znajduje się sekcja **skryptów** odpowiedzialna za kodowanie instrukcji dla naszego programu. Z części widocznej po jej lewej stronie będziemy „zabierać" potrzebne nam klocki i umieszczać je po prawej stronie w ten sposób, by kolejne elementy łączyły się ze sobą w sekwencje. Bloczki są pogrupowane tematycznie i kolorystycznie w zależności od swoich funkcji. To duże udogodnienie, które znacznie przyspiesza zaznajomienie się ze Scratchem oraz wyszukiwanie. Klikając prawym przyciskiem, możemy też usunąć lub zduplikować klocek.

Pierwszy program w Scratchu

Słowo programowanie przyprawia o gęsią skórkę i odrzuca od komputera. Minęły już jednak te czasy, w których programy tworzyło się na czarnych ekranach komputerów z białym tekstem. Tworzenie aplikacji jest naprawdę proste i naprawdę przyjemne. Nie trzeba też mieć szczególnych uzdolnień informatycznych, by bezproblemowo rozpocząć swoją przygodę z programowaniem. To po prostu bardzo dobra i wciągająca zabawa. Przekonajmy się o tym.

1 Najpierw przeciągamy bloczek **przesuń o... kroków** do sekcji skryptów. Jeśli teraz klikniemy na ten bloczek, to kot się poruszy. Odległość, jaką będzie przechodził zwierzak, zależy od wartości wpisanej w białe pole.

2 Przechodzimy do grupy fioletowych klocków o nazwie **Dźwięk**. Znajdujemy bloczek **zagraj bęben** i dołączamy go do poprzedniego. Klikając teraz już na dwa klocki, sprawimy, że wcześniejszemu ruchowi będzie towarzyszył odgłos bicia w bęben.



UWAGA!

Jeśli nie jesteśmy pewni, do czego służy dany klocek, możemy kliknąć na niego prawym przyciskiem myszy i wybrać **Pomoc**. Zobaczymy wtedy opis działania elementu.

3 Na fioletowym klocku znajduje się lista rozwijalna. Możemy na nią kliknąć, a następnie wybrać inny odgłos bębna.

4 Klikamy prawym przyciskiem myszy na element odpowiedzialny za przesuwanie postaci i wybieramy duplikuj. Powielają się oba klocki z naszego skryptu. Dołączamy je do poprzednich dwóch.

UWAGA!

Duplikowanie w Scratchu działa zawsze „w dół". Jeśli nasza instrukcja zawiera pięć połączonych bloczków, a my klikniemy na przykład na trzeci i wybierzemy opcję duplikuj, to powieli się bloczek trzeci, czwarty i piąty.

5 W pierwszym z nowo dodanych elementów zmieniamy liczbę klocków na wartość -10. W drugim – wybieramy inny rodzaj bębna. Możemy teraz uruchomić program i zobaczyć, co się będzie działo.

6 A gdybyśmy teraz zechcieli tak przerobić program, by raz włączony działał aż do momentu, gdy powiemy stop? Da się to zrobić bardzo prosto. Wystarczy w grupie żółtych bloczków znaleźć ten oznaczony jako **zawsze** i opleść nim nasze dotychczasowe instrukcje tak, jak na obrazku. Powstanie w ten sposób pętla. (Pętle możemy budować z żółtych bloczków przypominających wyglądem literę C – **zawsze**, **powtórz ... razy** i **powtarzaj aż**. Wewnątrz nich możemy umieszczać inne klocki, po prostu przenosząc je i upuszczając. Więcej o pętlach przeczytamy w dalszej części rozdziału).

7 Teraz, by przerwać działanie naszego programu, klikamy na czerwony znak stop w prawym górnym rogu ekranu.

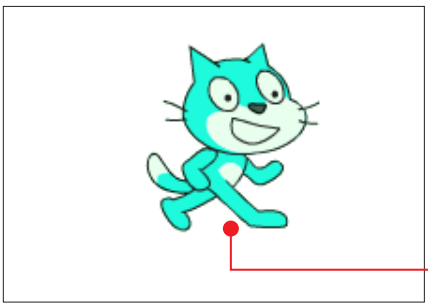
UWAGA!

Bloczki kiedy kliknięto powinny znajdować się w każdym z tworzonych programów. Inaczej użytkownicy włączający nasz skrypt mogą mieć problem z jego uruchomieniem.

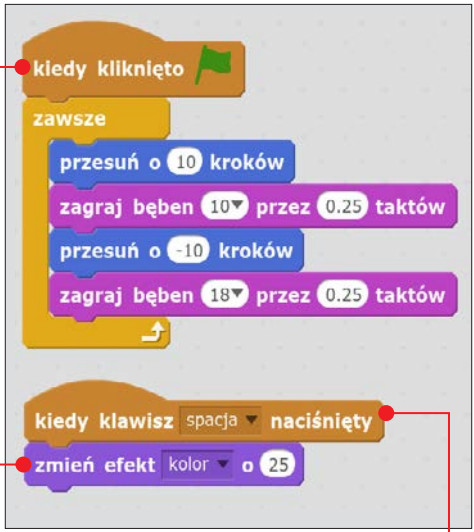
8 Z grupy ciemnożółtych klocków wybieramy ten o nazwie **kiedy kliknięto** i wstawiamy go na samą górę instrukcji.

Od teraz, gdy będziemy chcieli rozpocząć taniec kota, wystarczy, że klikniemy na zieloną flagę nad oknem z wykonywanym programem.

9 Teraz sprawimy, że nasz program będzie wykonywał dwie czynności jednocześnie. Przeciągamy bloczek **zmień efekt**, który znajduje się na zakładce **Wygląd**, ale nie dodajemy go do stworzonego wcześniej skryptu, tylko układamy oddzielnie.



10 Wstawiamy do naszego kodu jeszcze bloczek, **kiedy klawisz... naciśnięty**. Umieszczamy go nad bloczkiem zmieniającym efekt. Możemy, według własnego uznania, zmieniać klawisz, który będzie zmieniał wygląd (kolor) naszego kota. Włączamy program i sprawdzamy, co się stanie, gdy wciśniemy spacja. Właśnie skończyliśmy tworzyć nasz pierwszy program. Nawet tak prosty skrypt jak ten pokazuje, że środowisko Scratch ma bardzo duży potencjał. Widać również, że kod programu jest bardzo czytelny i zrozumiały także dla osób, które nie miały wcześniej kontaktu z programowaniem.



Strona projektu – co tu mogę znaleźć

Kiedy nasz program działa już dobrze, przychodzi pora na podzielenie się nim z szerszą publicznością. Nie tworzymy przecież naszych projektów dla samej idei tworzenia. Wręcz przeciwnie. Programuje się przede wszystkim dla innych ludzi, a informacja zwrotna od nich jest bardzo cenna. Powinna być dla nas motywacją do dalszej pracy. Opinie innych użytkowników pokażą, czego jeszcze brakuje naszym aplikacjom, dzięki czemu będziemy stawali się jeszcze lepszymi programistami.

Scratch pokazuje reakcje na nasz opublikowany program na specjalnej stronie zwanej stroną projektu.

Okno Scratcha

GŁÓWNE MENU PROGRAMU

Tu możemy zapisać ułożony przez nas program, pobrać go na swój komputer czy też wgrać projekt aplikacji stworzonej w starszej wersji Scratcha. Jeśli potrzebujemy więcej miejsca na bloczki, przełączamy się na układ małej sceny.

NARZĘDZIA DO SZYBKIEJ EDYCJI DUSZKA

Od lewej: duplikowanie, wycinanie, powiększanie i pomniejszanie. Wystarczy wybrać narzędzie, a następnie kliknąć na wybranego duszka widocznego w obszarze wykonywania programu.

Tutaj wpisujemy nazwę naszego programu.

Tu, na ekranie wykonywania aplikacji, możemy podejrzeć, jak ona działa.

Uruchamianie i zatrzymywanie skryptu dla wszystkich duszków. Zieloną flagą uruchomimy skrypt, a czerwonym znakiem stop zatrzymamy działanie całego programu.

Tu możemy podejrzeć aktualne współrzędne wskaźnika myszy, który znajduje się nad ekranem wykonywania programu.

Wybór sceny (czyli tła) dla wykonywanego skryptu. Domyślnie jest to biała plansza. Nowe sceny możemy: wgrywać z bazy Scratcha lub z naszego dysku, rysować samemu albo fotografować swoją kamerą internetową.

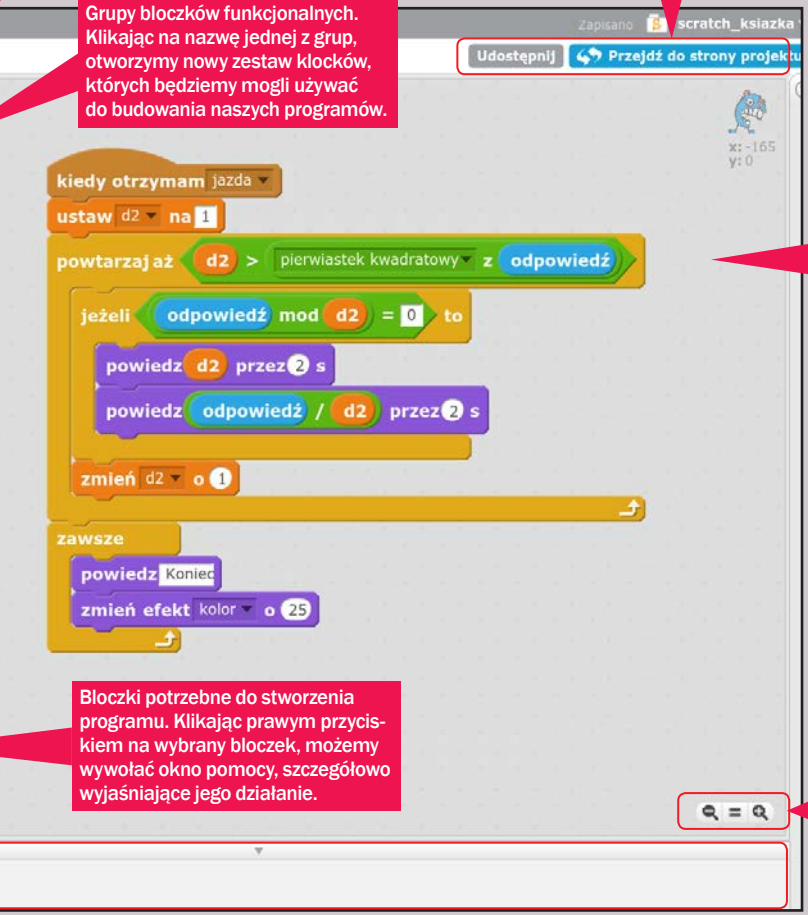
Wybieranie duszka, którego w danym momencie chcemy programować. Trzeba pamiętać, że nasz program może mieć kilka aktywnych duszków, a każdy może być inaczej zaprogramowany. Sekcja skryptów będzie się zmieniała w zależności od wybranego w danym momencie duszka.

W plecaku możemy przechowywać kostiumy, tła i skrypty od innych użytkowników. Możemy ich potem użyć w naszym programie.

Zakładki pozwalające na zmianę zawartości sekcji. Domyślnie wyświetla się zakładka **Skrypty**. Klikając na zakładkę **Kostiumy/Tła**, możemy dokonywać zmian w wyglądzie swoich czterech duszków czy scen. Zakładka **Dźwięki** pozwoli nam na dodanie do projektu odgłosów, które następnie mogą być wykorzystywane w działaniu programu.

Tymi przyciskami możemy udostępnić swój projekt, by mogli go podziwiać internauci z całego świata.

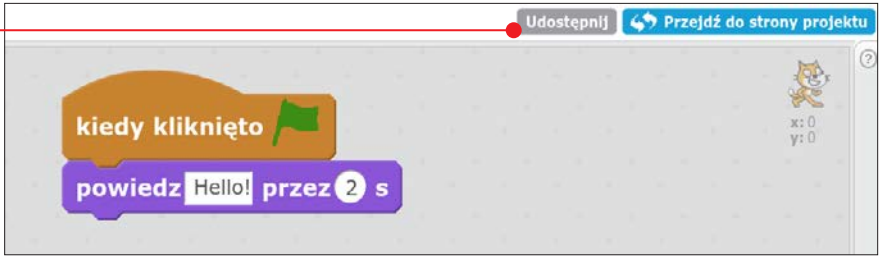
Grupy bloczków funkcjonalnych. Klikając na nazwę jednej z grup, otworzymy nowy zestaw klocków, których będziemy mogli używać do budowania naszych programów.



Tu należy przeciągać bloczki, by stworzyć swój program. Możemy łączyć ze sobą dowolną liczbę pasujących do siebie bloczków, tworząc ich segmenty. Usuwanie niepotrzebnego bloczka polega na kliknięciu na niego prawym przyciskiem myszy i wybraniu polecenia **Usuń**. W ten sam sposób można je duplikować.

Bloczki potrzebne do stworzenia programu. Klikając prawym przyciskiem na wybrany bloczek, możemy wywołać okno pomocy, szczegółowo wyjaśniające jego działanie.

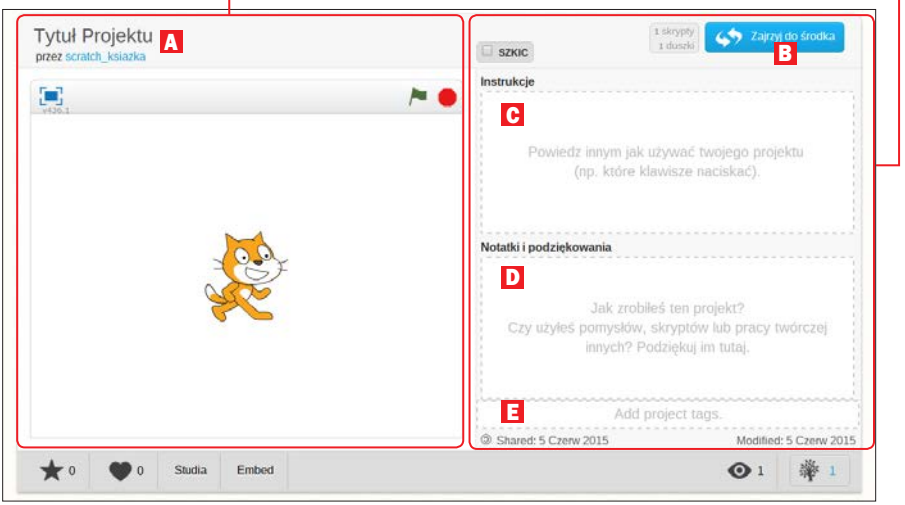
Tymi przyciskami możemy pomniejszać, przywracać do ustawień domyślnych lub powiększać bloczki naszego projektu.



1 Aby zebrać informacje zwrotne, najpierw musimy opublikować projekt – na przykład ten, który stworzyliśmy w poprzedniej poradzie. Wystarczy kliknąć na szary przycisk z napisem **Udostępnij** znajdujący się nad sekcją Skryptów. Przycisk zniknie, a strona automatycznie przeładuje się na gotową do uzupełnienia stronę projektu.

2 Strona projektu dzieli się na dwa podstawowe obszary. Pierwszy z nich zawiera opublikowany program wraz z informacjami o nim. Druga część to informacje na temat naszej pracy wysyłane przez innych użytkowników. Znajdują się tam zarówno komentarze innych twórców, jak i informacja o ich przynależności do studiów, czyli tematycznych grup programów.

Na stronie projektu można:
- Zmienić tytuł swojego programu **A**.
- Przejść do trybu edycji bloczków przez kliknięcie na niebieski przycisk z napisem **Zajrzyj do środka** **B**.



- Dodać **Instrukcje użytkowania projektu** C (na przykład Żeby zmienić kolor kota, wciśnij spację).
- Poniżej pola z instrukcjami znajduje się inne D, w którym możemy wpisać krótką historię powstawania programu oraz podziękowania dla osób, które pomogły w jego tworzeniu.
- Trzecie pole z prawej strony E należy uzupełnić tagami, czyli słowami kluczowymi, które najlepiej pasują do projektu. Przykładowo: jeśli stworzyliśmy kalkulator, to odpowiednim tagiem jest wyraz **matematyka**. Im więcej słów kluczowych, tym łatwiej będzie wyszukać naszą aplikację.

U dołu, pod programem, znajduje się szara belka z ikonami.
Znajdujące się na niej przyciski to:
A Polubienie projektu.
B Ukochanie aplikacji.
C Dodanie projektu do jednego ze swoich studiów.
D Osadzenie programu na swoim koncie na Twitterze, Facebooku i, co najważniejsze, możliwość skopiowania kodu HTML i opublikowania okna aplikacji na swojej stronie WWW.
E Ikona przedstawiająca oko podaje liczbę odsłon programu.
F Ikona drzewa pokazuje, ile razy nasza aplikacja została zremiksowana, czyli przekształcona lub użyta do stworzenia bardziej zaawansowanego skryptu.

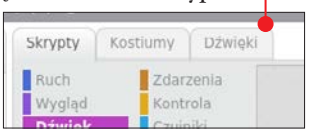
U dołu strony projektu znajduje się panel, w którym osoby z całego świata będą mogły wyrażać swoje opinie o naszej pracy, a my będziemy mogli im odpowiadać. Jeśli będziemy systematycznie pracować na swoim koncie, zauważymy, że wkrótce zaczną się tu toczyć żywiołowe dyskusje.

Scratch i muzyka

Programy tworzone w Scratchu cechują się prostym sposobem pracy z dźwiękiem. Efekty, które są trudne do osiągnięcia w innych środowiskach, tu mamy na wyciągnięcie ręki. Dźwięki pozwolą nam tworzyć interesujące animacje czy też ulepszyć działanie naszych aplikacji.
Bloczki odpowiedzialne za działanie dźwięku w Scratchu są jasnofioletowe i oznaczone napisem **Dźwięk**. Jest ich sporo, ale na szczęście dzięki ich nazwom od razu można domyślić się, jak działają. Warto zwrócić szczególną uwagę na bloczki **zagraj dźwięk...** oraz **zagraj dźwięk... i czekaj**. Drugi z nich sprawia, że ani-

macja czeka, aż skończy się podkład muzyczny, i dopiero potem przechodzi do wykonywania kolejnej czynności – bardzo dobre rozwiązanie, gdy chcemy dodawać efekty dźwiękowe dla duszków. Pierwszy włącza dźwięk i od razu zaczyna wykonywać resztę skryptu – można to wykorzystać na przykład do dodania tematu muzycznego działającego w tle programu. Obydwa wspomniane bloczki pozwalają wybrać wykonywany dźwięk (domyślnie odgłos miauczenia) z bazy programu. Co interesujące – bazę dźwięków możemy dowolnie rozbudowywać. Możemy wykorzystywać zarówno dźwięki dostarczone przez twórców Scratcha, jak i samemu nagrywać własne. Należy przy tym pamiętać, że im więcej dźwięków będzie w bazie Scratcha – tym wolniej program może się ładować. Aby dodać nowe dźwięki do naszego programu, musimy:

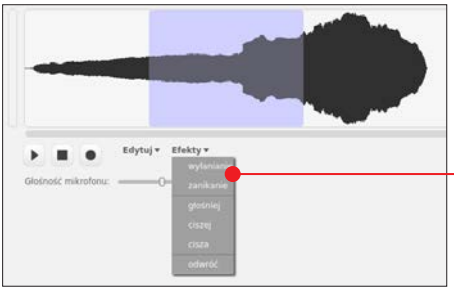
1 Włączyć zakładkę **Dźwięki**, widoczną nad ekranem skryptów.



2 Pod napisem **Nowy dźwięk** widzimy trzy ikony. Oznaczają one odpowiednio:

- **A** importowanie nagrań z bazy serwisu Scratch,
- **B** nagranie swojego dźwięku za pomocą mikrofonu w komputerze,
- **C** przesłanie pliku z dźwiękiem z komputera do programu (w formacie MP3 lub WAV).

W zakładce poświęconej dźwiękom bez problemu da się edytować, łączyć i usuwać fragmenty plików muzycznych – znajdziemy te opcje po kliknięciu na napis **Edytuj**. Możemy też, korzystając z listy rozwijalnej **Efekty**, zmienić działanie naszego nagrania, na przykład poprzez odwrócenie go w czasie czy stopniowe wyciszanie w miarę trwania. Praca w tym edytorze jest bardzo prosta, przypomina pracę w edytorze tekstowym. Musimy tylko zaznaczyć fragment, a następnie na jednej ze wspomnianych list wskazać interesującą nas czynność. Spróbujmy teraz sami ułożyć poniższy skrypt.



Zmienne

Programista musi doskonale umieć posługiwać się takimi narzędziami, jak zmienne i tablice. Jest tak z prostego powodu – każdy bardziej zaawansowany program operuje na danych, a te są właśnie przechowywane przez wspomniane obiekty. Spokojnie, nie ma w tym nic trudnego. Trochę praktyki i sami będziemy się dziwić, w jaki sposób dotąd mogliśmy się obchodzić bez tych funkcji.

Zacznijmy od zmiennych. Możemy je sobie wyobrazić jako pojemniki na przedmioty. Te pojemniki możemy w działaniu programu przenosić i wykorzystywać w wielu miejscach – wypisywać, porównywać czy wykonywać operacje matematyczne. Rzeczami, które będziemy w tych pojemnikach przenosić, będą informacje, czyli liczby i tekst.

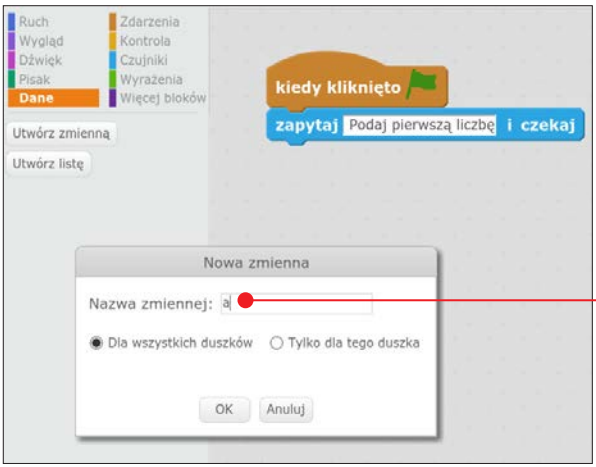
Teraz trochę praktyki. Stworzymy program obliczający wynik mnożenia dwóch liczb.

1 W nowym projekcie Scratcha standardowo zaczynamy od dodania bloczka inicjującego działanie skryptu, klikając na zieloną flagę. Dalej dołączamy bloczek **zapytaj... i czekaj**. Znajdziemy go w grupie jasnoniebieskich bloczków.

UWAGA!

Zmienne są bardzo ważne w programowaniu. Zaczniemy ich już używać w następnym rozdziale, ale największe znaczenie będą miały, gdy wspólnie zajmiemy się programami i grami w rozdziałach 4 i 5.

2 Pole tekstowe wewnątrz bloczka wypełniamy tekstem **Podaj pierwszą liczbę**. Będzie to komunikat, który pokaże się użytkownikowi pracującemu z naszym programem. Będzie widoczny aż do momentu, gdy użytkownik poda informację zwrotną.



3 Przejdźmy teraz do kategorii bloczków o nazwie **Dane**. To ta pomarańczowa.

4 Klikając na przycisk **Utwórz zmienną**, dodamy nową zmienną. Nazwijmy ją **a**.

5 Dołączmy teraz do programu pomarańczowy klocek **ustaw... na...**.

Z rozwijanej listy wybieramy zmienną **a**. Drugie puste pole zapełniamy z kolei jasnoniebieskim bloczkiem z napisem **odpowiedź**. Jest to specjalna zmienna tymczasowa, przenosząca treść odpowiedzi udzielonej przez użytkownika programu.

6 W analogiczny sposób dodajmy pytanie o drugą liczbę.

7 Od teraz nasz program potrafi już zbierać i przechowywać dane. Czas je wykorzystać. Używając dwóch ciemnofioletowych bloczków, sprawiających, że kot będzie mówił, zielonego bloczka służącego do mnożenia dwóch elementów oraz pomarańczowych zmiennych, dołączamy następujący kod do naszego programu.

8 Włączamy program. Wszystko powinno działać.

Nasz program nie jest oczywiście doskonały. Można łatwo go zepsuć, wpisując odpowiedź na któreś z pytań jako literę czy wyraz. By to naprawić, musielibyśmy zająć się tak zwaną obsługą błędów. Nauczymy się tego w dalszej części książki.

Listy

Listy w Scratchu to narzędzie, które pozwala przechowywać wiele informacji w jednym miejscu. To taka zmienna, która przechowuje w sobie inne zmienne. Osoby, które miały już do czynienia z programowaniem, z pewnością spotkały się z czymś takim jak **tablice**. Listy w Scratchu to dokładnie to samo.

Dodając kolejne elementy do listy, nie musimy się obawiać o jej przepełnienie. Lista może mieć dowolnie dużo wartości. Każda z nich będzie

UWAGA!

Bloczki powiedz i pomyśl są dobrym rozwiązaniem, gdy chcemy komunikować się z użytkownikami korzystającymi z naszej aplikacji.

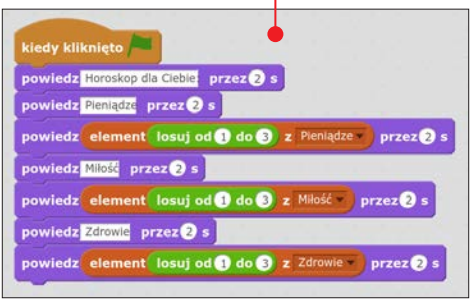
miała swoje miejsce, po którego numerze będziemy mogli ją później wywołać. Działanie list można zaprezentować w bardzo ciekawy sposób. Z ich pomocą stworzymy za chwilę program przepowiadający przyszłość. Dowiemy się również, w jaki sposób losować wartości z dowolnie wybranych zbiorów. Do dzieła!

1 Zacznijmy od stworzenia trzech list o nazwach: **Miłość**, **Pieniądze** i **Zdrowie**. Robimy to w analogiczny sposób, jak tworzyliśmy wcześniej zmienne, z tą różnicą, że w zakładce **Dane** klikamy na przycisk **Utwórz listę**.

2 Teraz pora na zapełnienie wszystkich list danymi, tak by program miał bazę wróżb do naszego horoskopu. Stosowna instrukcja powinna wyglądać jak ta.

3 To, czy dobrze wstawiliśmy elementy do naszych list, będziemy mogli podejrzeć w oknie programu. Jeśli instrukcji w którejś z list jest zbyt wiele, możemy je łatwo usunąć, korzystając ze stosownych bloczków. Gdy wszystko jest poprawnie, podgląd list nie będzie dalej potrzebny. Możemy go wyłączyć, usuwając zaznaczenia z odpowiednich pól w zakładce **Dane**.

4 Teraz zajmiemy się częścią programu, odpowiedzialną za nasz horoskop. Ułożmy go w taki sposób.

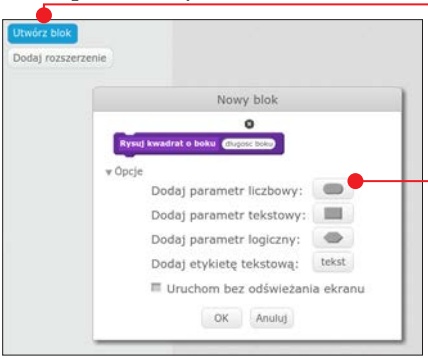


Po raz pierwszy używamy tu zielonego bloczka **losuj... od... do...**. W naszym programie będzie wybierał jeden element spośród trzech w każdej z list.

5 Możemy już kliknąć na zieloną flagę, by uruchomić stworzoną przez nas aplikację. Za każdym razem nasz duszek – kot – będzie podawał nam inną przepowiednię.
Program możemy teraz dowolnie modyfikować – dodawać nowe listy lub ich elementy. Poeksperymentujmy teraz na własną rękę.

Tworzenie swoich bloczków

Od najnowszej wersji Scratcha istnieje możliwość tworzenia własnych bloczków. Osoby mające już wcześniej do czynienia z programowaniem, mogą je utożsamiać z procedurami. To bardzo przydatna funkcja, szczególnie jeśli planujemy tworzyć zaawansowane programy. Nowe klocki pozwolą zaoszczędzić nam dużo czasu, zwłaszcza jeśli w naszym programie pewna czynność powtarza się kilka razy. Spróbujemy stworzyć teraz bloczek, który będzie rysował kwadrat o podanej przez użytkownika długości boku. Użyjemy w tym celu dobrze znanych już poleceń, a także poznamy nowe – służące do tworzenia pętli oraz rysowania.



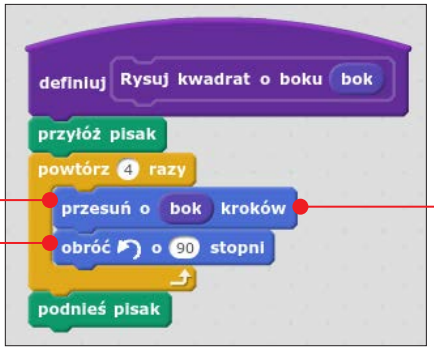
1 Na ekranie tworzenia programu przechodzimy do zakładki **Więcej bloków**, a następnie klikamy na **Utwórz blok**.

2 Pojawi się nowe okno z fioletowym klockiem z pustym wnętrzem. Zapełniamy je poleceniem, które bloczek będzie wykonywał (na przykład **Rysuj kwadrat o boku**).

3 Rozwijamy listę opcji i klikamy na przycisk **Dodaj parametr liczbowy**. W jego wnętrzu wpisujemy wyraz **bok** – będzie to zmienna, która pozwoli nam opisać działanie naszego programu.

4 Pora zdefiniować działania naszego klocka. W części skryptowej pojawił się nowy, fioletowy bloczek. Warto zauważyć w nim pojawienie się zmiennej **bok**,

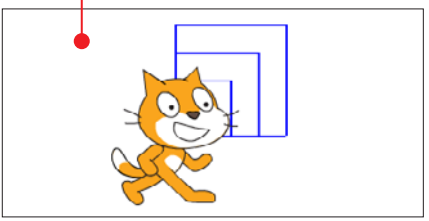
którą możemy „wyciągnąć” i użyć w naszym programie.



5 Ułożenie kodu nie powinno przysporzyć kłopotów. Nasz kot musi cztery razy podejść do przodu ●, za każdym razem obracając się o 90 stopni w wybranym kierunku ●. W wypadku takich powtarzających się czynności idealnie sprawdzą się pętle. My skorzystamy z **powtórz... razy**. Wszystkie instrukcje zawarte w tej pętli będą się powtarzały wymienioną w nagłówku bloczka liczbę razy.

6 Gdy cały skrypt będzie już gotowy, czas na jego wypróbowanie. Ułóżmy kod taki jak ten ●, by zobaczyć, czy wszystko działa.

7 Efekt powinien wyglądać następująco ●.



Pętle i instrukcje warunkowe

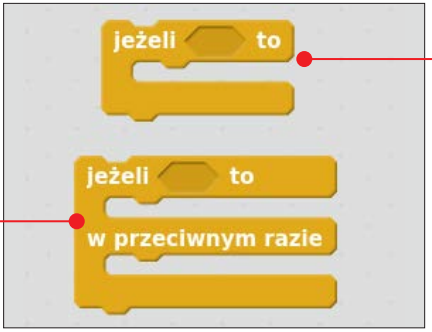
Pętle to narzędzia programistyczne, pozwalające powtarzać wielokrotnie czynność, jaką ma zajmować się program. Mogą one mieć albo określoną liczbę powtórzeń (powtórz 10 razy), albo działać, aż zostanie spełniony określony warunek (powtarzaj, aż zmienna **x** będzie większa od 10). Pętle pozwalają na znaczne zmniejszenie liczby elementów, z jakich musielibyśmy stworzyć nasz skrypt.

Scratch oferuje trzy rodzaje pętli, których możemy używać do budowania programów. Będą one powtarzały instrukcje, które w nich umieścimy. Znajdują się w grupie żółtych bloczków ● o nazwie **Kontrola**:

- **powtórz... razy** – pętla będzie miała tyle przejść, ile wpiszemy w pole tekstowe,
- **zawsze** – pętla raz uruchomiona będzie wykonywała instrukcje w niej zawarte tak długo, aż zostanie naciśnięty czerwony przycisk **Stop** u góry ekranu wykonywania aplikacji,
- **powtarzaj aż...** – instrukcje zawarte w tej pętli będą tak długo powtarzane, aż warunek postawiony na początku zostanie spełniony.

Stosowanie pętli przećwiczymy w następnym rozdziale, poświęconym tworzeniu rysunków w Scratchu.

Instrukcje warunkowe pozwalają natomiast na wykonanie lub niewykonanie pewnej części programu w zależności od tego, czy twierdzenie w niej zawarte będzie prawdziwe czy fałszywe. Takich instrukcji będziemy używać w rozdziale **4** do stworzenia prostego kalkulatora albo do obsługi tablicy rekordów w grach. W Scratchu mamy do dyspozycji dwie instrukcje warunkowe, które wyglądają bardzo podobnie. Tak jak pętle możemy



je znaleźć w grupie bloczków o nazwie **Kontrola**.

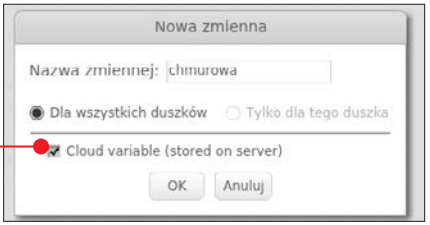
Pierwsza z nich – **jeżeli...** – pozwala na wykonanie instrukcji w zależności od spełnienia postawionego warunku.

Druga to rozszerzenie pierwszej o to, co ma się dziać, jeśli warunek nie będzie spełniony.

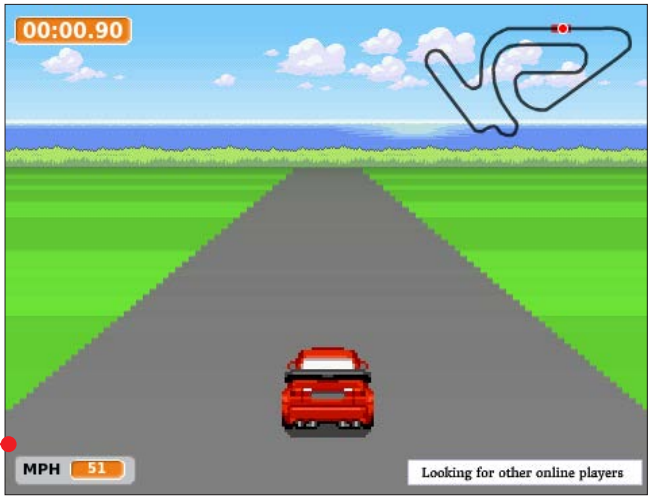
Instrukcja warunkowa będzie działała poprawnie, jeśli umieścimy w sześciokątnym polu wewnątrz niej odpowiedni warunek, czyli taki, któremu jednoznacznie da się przypisać miano prawdy lub fałszu (taki warunek to na przykład odpowiedź = 0).

Zmienne chmurowe

W nowym Scratchu możemy używać **Danych w chmurze**. To specjalny rodzaj zmiennej – może zapisywać dane na serwerach Scratcha i nie jest kasowana po każdym uruchomieniu programu. Aby korzystać z tej zmiennej, musimy spełnić tylko jeden warunek – nasze konto musi być aktywne. Jeśli będziemy stale publikowali nowe projekty, komentowali prace innych i oceniali je, to po pewnym czasie, dodając nową zmienną, zobaczymy nowe pole, które będziemy mogli zaznaczyć. Jeśli to zrobimy, dodawana zmienna stanie



się **zmienną chmurową**. Takie zmienne mają jednak pewne ograniczenie – mogą przechowywać tylko wartości związane z liczbami całkowitymi. Nie możemy w nich zatem zapisać na przykład liter i wyrazów. Możliwości, jakie otrzymujemy, i tak są jednak ogromne. Zmienne chmurowe doskonale nadają się do tworzenia gier, w których możemy rywalizować o najwyższy wynik z osobami z całego świata. Bardzo proste jest również tworzenie ankiet, w których użytkownicy odpowiadają na pytanie zadane przez twórcę aplikacji, a sam program podsumowuje odpowiedzi na wykresach słupkowych.



Korzystanie z dołączonej płyty

Do książki jest dołączona płyta – oto jej menu. Znajdują się na niej materiały pomocnicze. Jest to zarówno sam edytor Scratch, jak i programy, takie jak środowisko Adobe Air czy aplikacja pozwalająca na rysowanie schematów podłączenia

różnych peryferiów do płytki Arduino, omawiana w rozdziale **7**.

Skrypty na płycie

Najważniejsze są jednak skrypty Scratcha – kody będące efektem wszystkich przedstawionych w książce wskazówek. Pliki te mają rozszerzenie .sb2 i wymagają do działania posiadania konta w serwisie Scratch lub zainstalowanego edytora Scratch offline. Skrypty znajdziemy na płycie w dziale **Materiały edukacyjne**. W menu płyty zostały opisane numerami rozdziałów książki, w których są omówione.

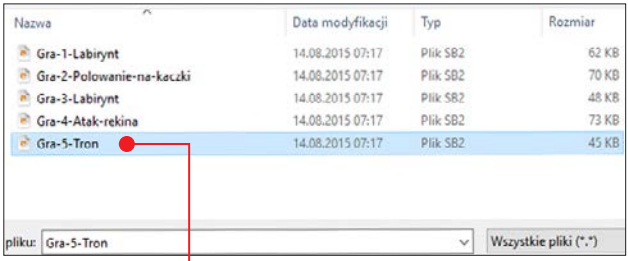
1 Aby skorzystać ze skryptów, w menu płyty klikamy na **Materiały edukacyjne**, a potem wybieramy kliknięciem rozdział.

2 Następnie klikamy na **Instaluj**. Archiwum ze skryptami zostanie automatycznie zapisane na pulpicie i rozpakowane do folderu o nazwie **Scratch**. Gotowe – możemy już korzystać z plików.

Uruchamianie plików Scratcha w edytorze offline

1 Uruchamiamy edytor Scratch offline.

2 W programie klikamy na **Plik** i na **Otwórz**, a następnie wskazujemy plik zapisany na dysku – w naszym przykładzie jest to skrypt gry Tron z rozdziału **5** umieszczony na pulpicie w folderze **Scratch**. Klikamy na przycisk **Otwórz**.

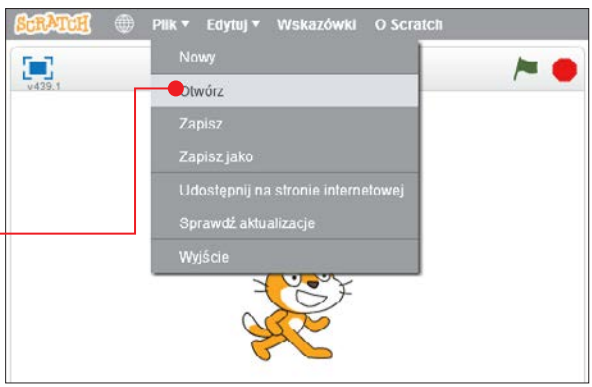


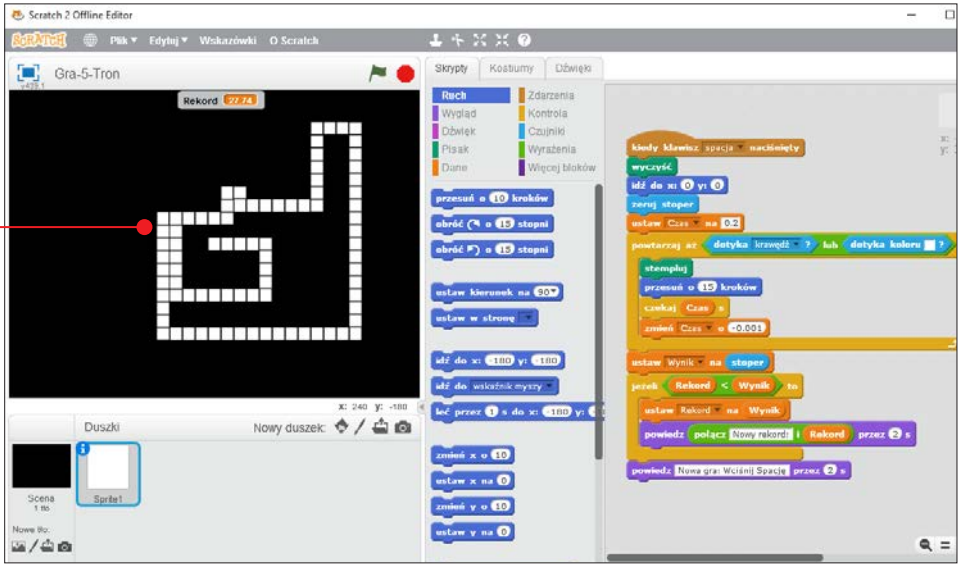
3 Aplikacja stworzona w Scratchu jest już otwarta i gotowa do użycia. W wypadku gry Tron, by zacząć grać, wciskamy klawisz spacja i kierujemy ruchem kwadratu za pomocą strzałek na klawiaturze. Im dłużej będzie nam się to udawało, tym lepiej (więcej w rozdziale **5**).

Uruchamianie plików Scratcha w edytorze online

1 Logujemy się w serwisie Scratch na swoje konto.

2 Otwieramy nowy projekt, klikając na **Stwórz** u góry okna.





3 W edytorze, na szarym pasku u góry, klikamy na **Plik** i na **Otwórz**, a następnie wskazujemy plik na dysku, na przykład **Gra-5-Tron.sb2** z rozdziału **5** zapisany na pulpicie. Klikamy ponownie na przycisk **Otwórz**.

KOŃCZENIE PRACY W SCRATCHU

■ W serwisie online wszystkie zmiany są zapisywane automatycznie. Nie musimy więc niczego zapisywać. Zakończenie pracy z projektem polega na zwyczajnym zamknięciu aktywnej karty przeglądarki.

■ W edytorze offline, żeby zapisać projekt, klikamy na **Plik**, **Zapisz** lub **Plik**, **Zapisz jako**. Następnie wskazujemy miejsce na dysku i ponownie klikamy na **Zapisz**.

UWAGA!

Pamiętajmy, żeby po zakończeniu pracy z serwisem internetowym Scratcha wylogować się z niego. Zadbamy w ten sposób o bezpieczeństwo naszych projektów i danych osobistych.

4 Podobnie jak w poprzednim przykładzie, możemy uruchomić grę i zacząć zabawę, wciskając klawisz spacja.

UWAGA!

Scratch pozwala na zapisanie projektów online, na internetowym koncie w serwisie albo lokalnie, na dysku komputera, w formie plików o rozszerzeniu .sb2. Pliki .sb2 można otwierać w edytorze online i offline.

ROZDZIAŁ 3

Rysujemy ze Scratchem

Animowane rysunki tworzone w Scratchu są bardzo efektowne. Przeczytajmy, jak je projektować

Jedną z podstawowych funkcji, których można używać w Scratchu, jest animacja rysowania fantastycznych wzorów. Duszek, podążając dowolnie wytyczoną ścieżką, może zostawiać za sobą ślad w postaci linii. W tym rozdziale nauczymy się rysować dowolne kształty. Najpierw stworzymy pierwszy rysunek krok po kroku. A dalej znajdziemy „zadania z rysowania" – z podpowiedziami i rozwiązaniami. O wiele więcej wzorów rysunków pobierzemy z KŚ+ (www.ksplus.pl). Gotowe skrypty do wszystkich zadań zostały zamieszczone na płycie dołączonej do książki oraz także w KŚ+.

DROGOWSKAZ

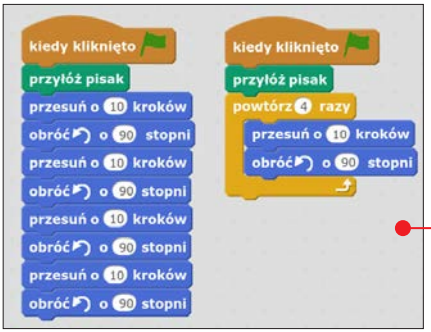
Jak rysować w Scratchu

Znamy już niezbędne podstawy potrzebne do rozpoczęcia pracy z bardziej zaawansowanymi skryptami. W tym rozdziale wykorzystamy zdobytą do tej pory wiedzę i trochę więcej się napracujemy. Dostaniemy zadania do samodzielnego wykonania (ale bez strachu – będą też podpowiedzi i opcjonalne rozwiązania). Będziemy tworzyć w Scratchu programy rysujące rozmaite wzory ●. Dzięki nim poćwiczymy umiejętność planowania i wyobraźnię. Będziemy bowiem musieli ułożyć odpowiednią strategię poruszania się duszka, tak by swoją ścieżką wykreślił odpowiedni kształt. (Takie zadania mogą wydać się znajome osobom korzystającym kiedyś z programów typu Logo, w Scratchu jednak nacisk kładzie się bardziej na układanie programu niż na zapisywanie kodu).

Dobre rady na początek

■ Duszek w Scratchu zawsze rysuje do krawędzi ekranu wykonywania programu. Skrypty, które zbliżają się do tego rejonu, mogą się źle rysować. Lepiej na początku tworzyć mniejsze rysunki, a gdy jesteśmy ich już pewni – zwiększać długości rysowanych kresek.

■ Dobrze mieć w polu skryptów „luźny” bloczek **wyczyść** ●. Gdy zrobimy niefortunną kreskę, a na pewno zrobimy ją nieraz, klikając na niego dwukrotnie, wyczyścimy wszystko,



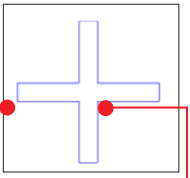
co zostało narysowane na ekranie, bez potrzeby uruchamiania naszego programu.

■ Starajmy się układać programy z użyciem jak najmniejszej liczby klocków. To, że program działa, to połowa sukcesu. Po każdym skończonym skrypcie zastanówmy się, czy nie dałoby się go zrobić prościej – „taniej”. Im „taniej”, tym lepiej ●.

■ Bloczki **ustaw kolor pisaka na...** i **zmień kolor pisaka o...** pozwalają na manipulacje barwą ścieżki, jaką zostawia nasz duszek. Pierwszy z nich ustawia jeden wybrany kolor, a drugi zmienia go o odcień, co ładnie wygląda użyte w pętli.

Te klocki w zupełności wystarczają do tworzenia zachwycających rysunków

Uczymy się rysować krok po kroku

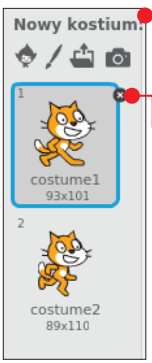


Wspólnie spróbujemy teraz stworzyć skrypt, który narysuje dla nas znak plus. Wzór będzie rysowany przez innego duszka niż domyślny – wybierzemy go z bazy Scratcha. Do stworzenia skryptu wykorzystamy informacje zdobyte w poprzednim rozdziale, szczególnie te dotyczące pętli. Zaczynamy rysować od środka.

Wybieramy i modyfikujemy duszka

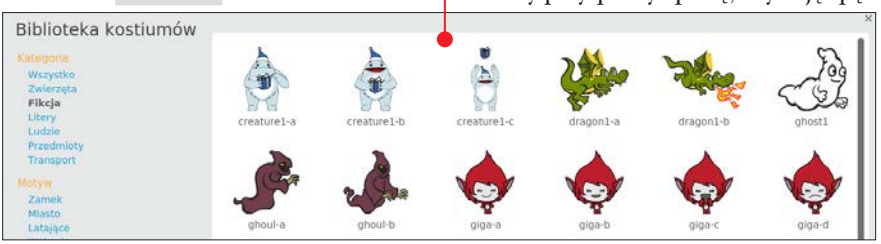
1 Otwieramy nowy projekt Scratcha online albo za pomocą edytora offline.

2 Aby zmienić duszka na innego, dostępnego w bazie Scratcha, przechodzimy do zakładki **Kostiumy**. Po lewej stronie zobaczymy panel z kostiumami (czyli duszkami).

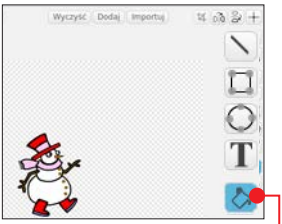


3 Niepotrzebny kostium możemy skasować z projektu, klikając na – nie będziemy go używać.

4 Pozostałego kota zmienimy na innego duszka. Klikamy w tym celu na widoczną nad kotem ikonę z małą postacią podpisaną **Wybierz kostium z biblioteki**.



5 Otworzy się nowe okno. Tu możemy wybrać nowego duszka. Pomocne mogą okazać się kategorie widoczne po lewej stronie.



6 Wybierzmy na przykład bałwanka. Zrobimy to, klikając na niego dwukrotnie myszą. Następnie, korzystając z edytora grafiki i zawartego tam narzędzia wypełniania, zmienimy kolor jego kapelusza i szala. Kostium jest już gotowy.

Kodujemy zachowanie duszka

1 Wracamy na zakładkę **Skrypty**. Przeciągnijmy do tego ekranu dwa bloczki – nasz program musi się uruchamiać po kliknięciu na zieloną flagę oraz zostawiać ślad trasy, jaką podąża duszek – to zapewni bloczek **przyłóż pisak**. Znajdziemy go w grupie bloczków **Pisak**.

2 Teraz zajmiemy się opisem drogi, jaką duszek ma podążać, żeby narysować znak plus. Tworząc rysunki w Scratchu, warto doszukiwać się powtórzeń. Dzięki nim możemy przyspieszyć pracę, używając pęt-

li. Znak plus składa się z czterech kształtów wyglądających jak kanciasta litera U. Musimy więc zakodować jedną literę U i wstawić ten kod do pętli powtarzającej się cztery razy.

UWAGA!

Pracując z rysunkami, warto umieścić w skryptach bloczek z grupy **Pisak** o nazwie **wyczyść**. Jeśli coś z naszym rysunkiem pójdzie nie tak, to dwukrotne kliknięcie na ten klocek wyczyści ekran wykonywania aplikacji.

3 By zakodować literę U, bałwan powinien przejść pewien odcinek do przodu – ustalmy, że 100 kroków – potem obrócić się o kąt 90 stopni w lewo, przejść krótszy odcinek, na przykład 30 kroków, znowu obrócić się w lewo o kąt prosty i przejść 100 kroków. Jedno ramię plusa już się rysuje.

4 Wszystkie wstawione bloczki z grupy **Ruch** umieszczamy w pętli, która będzie powtarzała się czterokrotnie. Musimy jeszcze dodać jeden bloczek przed końcem pętli. Ma on sprawiać, że duszek obróci się o 90 stopni w prawo – w ten sposób bałwan po pierwszym przejściu pętli ustawi się w kierunku do góry, po drugim przejściu – w lewo, a po trzecim – w dół. Gdybyśmy nie dostawili tego bloczka, przed rozpoczęciem następnej pętli duszek byłby źle ustawiony i nie narysowałby znaku dodawania.

UWAGA!

Oprócz bloczka **przyłóż pisak** jest też bloczek **podnieś pisak**, który zatrzymuje zostawianie śladu za duszkiem. Jeżeli jednak rysunek jest tworzony „jednym pociągnięciem", bez odrywania pisaka, wtedy ten bloczek jest niepotrzebny.

5 Możemy poeksperymentować z bloczkami z grupy **Pisak** – pozwalają na modyfikację śladu duszka. Możemy zmieniać kolor, odcień i grubość pisaka na dwa sposoby – ustawić konkretną wartość albo zaprogramować zmianę o określoną wartość. Interesująca jest ta druga możliwość – w pętli pozwoli na zmiany wyglądu śladu po każdym jej przejściu.

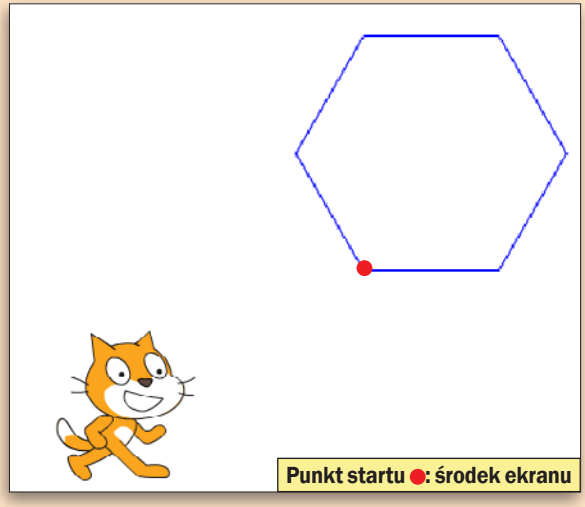
UWAGA!

Kolor pisaka należy zmieniać w zakresie wartości od 1 do 200. Kolory opisane liczbami większymi niż 200 to powtórzenia tych z przedziału 1–200.

6 Udoskonalmy kod, dodając bloczek **ustaw rozmiar pisaka na...**, z wpisaną wartością **3** na początku skryptu. Przeciągnijmy go pod bloczek **przyłóż pisak**.

7 Dodajmy też bloczek **zmień kolor pisaka o...** tak, żeby był ostatnim działającym w pętli. Wpiszmy do niego wartość **10**. Teraz kolor pisaka zmieni się nieco po każdym przejściu pętli.

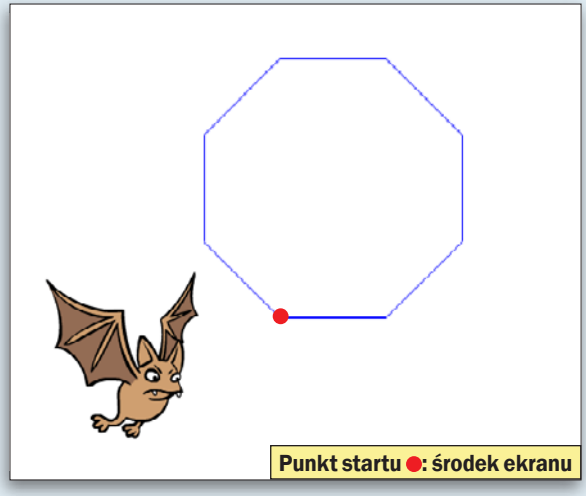
Rysunek 1: Sześciokąt foremny



Podpowiedzi
Rysunek 1

Kluczem do sukcesu jest odpowiedni kąt obrotu. Zauważmy, że każdy sześciokąt foremny możemy podzielić na sześć identycznych trójkątów równobocznych.

Rysunek 2: Ośmiokąt foremny



Podpowiedzi
Rysunek 2

Zadanie to jest jakby kontynuacją zadania pierwszego. Zacznijmy więc od wytypowania odpowiedniego kąta obrotu duszka.

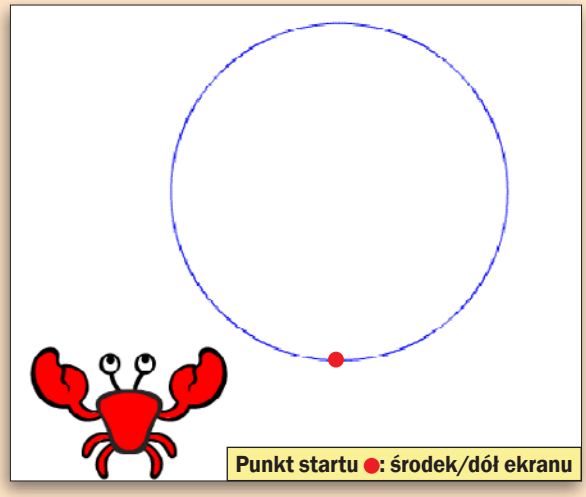
Rozwiązanie
Rysunek 1

Nasz duszek najpierw idzie przed siebie na określoną odległość (90 kroków), a następnie obraca się o 60 stopni. Akurat taki kąt sprawi, że narysowane odcinki będą do siebie pod kątem 120 stopni, czyli tak, jak ma być w sześciokącie foremnym. Ta czynność jest powtarzana sześć razy.

Rozwiązanie
Rysunek 2

Nasz duszek najpierw idzie określoną odległość przed siebie (80 kroków), a następnie obraca się o 45 stopni. Taki kąt sprawi, że narysowane odcinki będą tworzyły kąt 135 stopni, czyli tak, jak ma być w ośmiokącie foremnym. Ta czynność jest powtarzana osiem razy.

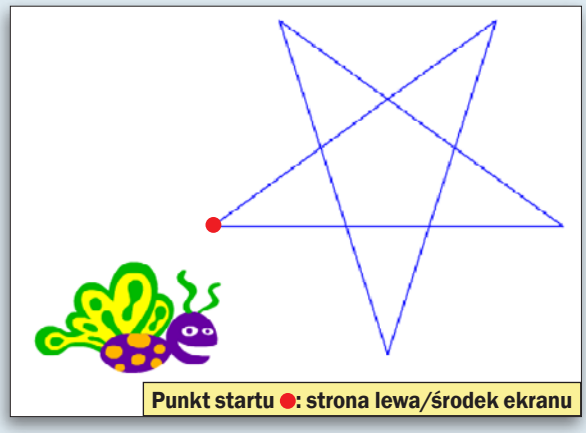
Rysunek 3: Okrąg



Podpowiedzi
Rysunek 3

Okrąg to tak naprawdę 360-stopniowy kąt foremny. Zadbajmy też o to, żeby bok tej figury był małą liczbą. Inaczej zabraknie nam pola do rysowania.

Rysunek 4: Pięciokąt gwiaździsty – pentagram



Podpowiedzi
Rysunek 4

Pentagram powstaje przez wyrysowanie wszystkich przekątnych pięciokąta foremnego. Kluczem jest znalezienie miary kąta między dwoma przekątnymi tej figury. Rysunek na kartce może bardzo pomóc.

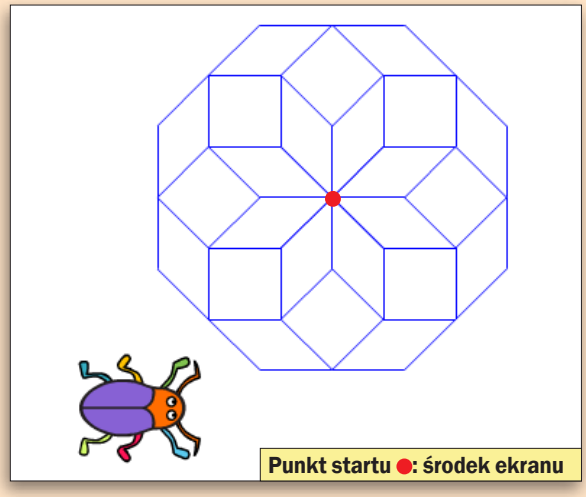
Rozwiązanie
Rysunek 3

Tworzymy pętlę, która 360 razy sprawi, że duszek pójdzie o 2 kroki przed siebie, po czym obróci się o 1 stopień. Gotowy rysunek da złudzenie okręgu.

Rozwiązanie
Rysunek 4

Gdy wewnątrz pięciokąta narysujemy wszystkie przekątne, w środku powstanie pentagram. Zauważmy, że kąt między przekątnymi wychodzącymi z dowolnego wierzchołka jest równy 36 stopni. Stąd obrót duszka musi być równy 144 stopnie (czyli 180 stopni – 36 stopni).

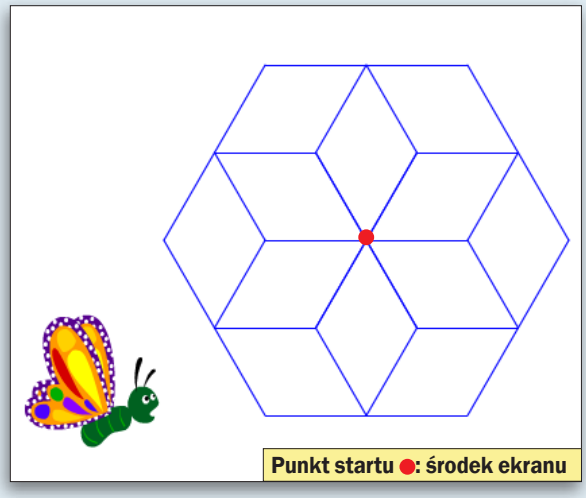
Rysunek 5: Parkietaż 1



Podpowiedzi
Rysunek 5

Ten rysunek powstał przez obracanie i rysowanie takiego samego ośmiokąta wokół jednego punktu. Postarajmy się rozwiązać problem, wstawiając pętlę do innej pętli.

Rysunek 6: Parkietaż 2



Podpowiedzi
Rysunek 6

Zadanie analogiczne do poprzedniego. Tu również należy zastosować pętlę w pętli. Ważne jest to, by zastosować w skrypcie odpowiednie kąty.

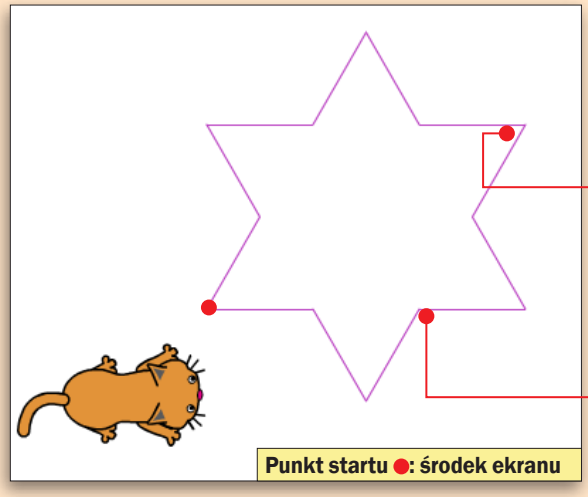
Rozwiązanie
Rysunek 5

Nasz robaczek rysuje ośmiokąt foremny. Gdy instrukcja odpowiedzialna za rysowanie się kończy, następuje wyjście do pętli nadrzędnej. Tam ma on już tylko jedno zadanie – obrócić się o kąt 45 stopni. Potem cały proces zaczyna się od początku.

Rozwiązanie
Rysunek 6

Rysowanie sześciokąta, które wykonuje się w wewnętrznej pętli, jest proste do wykonania. Po tej czynności nasz duszek powinien obrócić się o kąt 60 stopni, by linie następnego sześciokąta nałożyły się na linie poprzedniego.

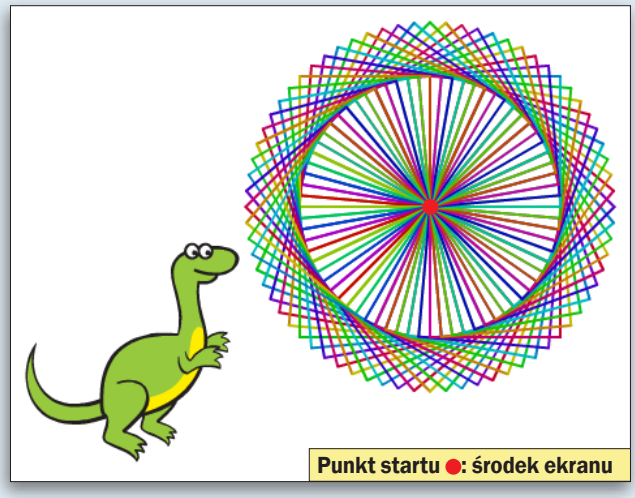
Rysunek 7: Gwiazda



Podpowiedzi
Rysunek 7

Gwiazda jest tworzona z sześciu elementów w kształcie litery **V**. Kąt wewnętrzny w ramieniu gwiazdy jest równy 60 stopni, a kąt zewnętrzny to 120 stopni.

Rysunek 8: Kolorowa rozeta



Podpowiedzi
Rysunek 8

Rozeta powstała poprzez obracanie kwadratu wokół jednego punktu. Kąt obrotu jest dość mały. Dodatkowo po narysowaniu każdego kwadratu zmienia się kolor następnego.

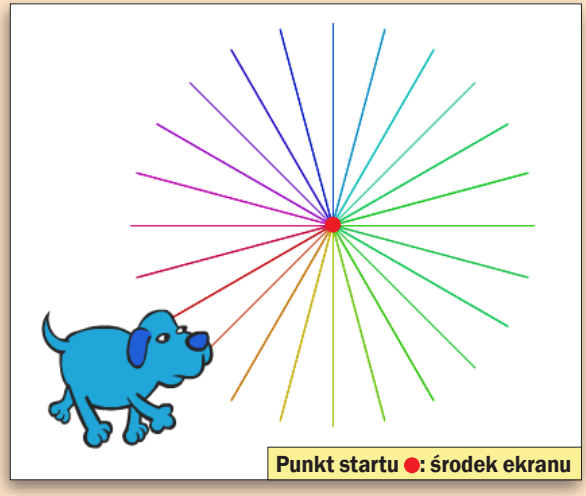
Rozwiązanie
Rysunek 7

Literę **V** tworzymy, dwukrotnie przesuwając duszka i obracając go między tymi ruchami o kąt 60 stopni. Następnie odbijamy duszka obrotem w przeciwnym kierunku o kąt 120 stopni. Sześciokrotne powtórzenie tej akcji sprawi, że otrzymamy gwiazdę.

Rozwiązanie
Rysunek 8

Żeby rysunek był ładny, zwiększamy grubość kreski i ustawiamy jasnozielony kolor pisaka. Następnie wewnątrz pętli głównej układamy następną, rysującą kwadrat o boku 100 kroków. Po końcu każdego przejścia tej pętli duszek obróci się o kąt 5 stopni i zmieni kolor pisaka. Ważne jest tu ustalenie liczby powtórzeń pętli głównej. W naszym przykładzie wynosi 72, gdyż tyle obrotów o kąt 5 stopni da łącznie obrót o kąt 360 stopni.

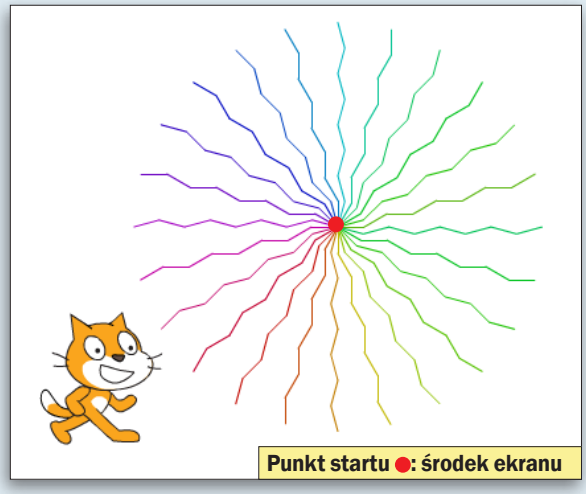
Rysunek 9: Promienie



Podpowiedzi
Rysunek 9

Rysunek powstał przez rysowanie linii i obrót duszka o stały kąt. Po każdej z tych czynności duszek wraca do centrum obrazka.

Rysunek 10: Krzywe promienie



Podpowiedzi
Rysunek 10

To zadanie jest rozwinięciem poprzedniego, z tą różnicą, że każda z kresek jest łamaną. Możemy to osiągnąć, tworząc odpowiednią pętlę.

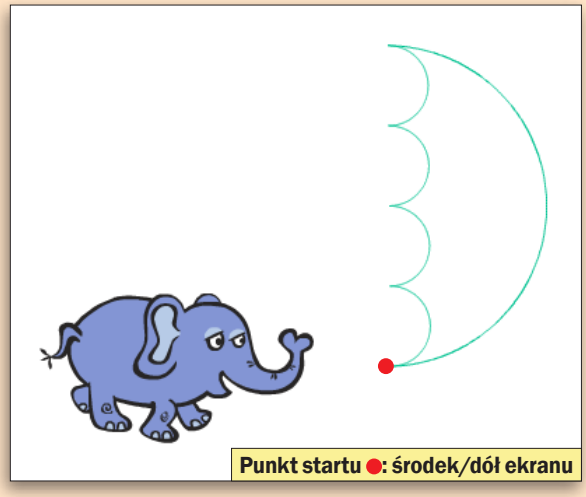
Rozwiązanie
Rysunek 9

Sam proces rysowania nie wymaga dłuższych wyjaśnień. Po jego ukończeniu będziemy mogli wrócić do punktu centralnego obszaru wykonywania programu, dokładając wewnątrz pętli bloczek **idź do x:0 y:0**.

Rozwiązanie
Rysunek 10

Zygzaki tworzymy za pomocą pętli sprawiającej, że duszek obraca się o dany kąt, idzie naprzód, a następnie odbija w drugą stronę o identyczny kąt i idzie przed siebie. Po kilku takich powtórzeniach duszek powinien powrócić do centralnego punktu ekranu aplikacji. W momencie powrotu musimy podnieść pisak – inaczej narysujemy prostą linię. Teraz musimy tę samą czynność powtarzać, obracając duszka o odpowiedni kąt.

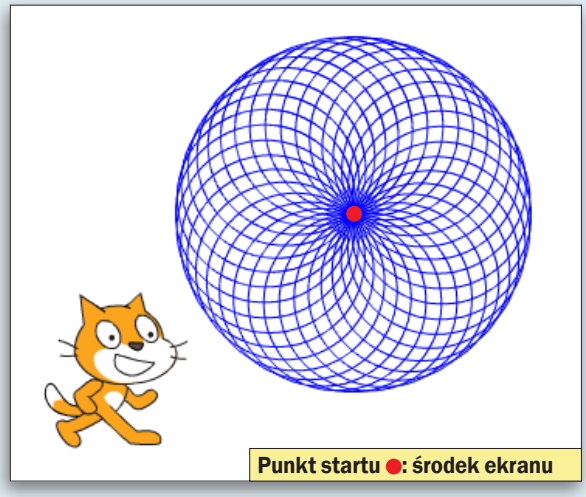
Rysunek 11: Piła



Podpowiedzi
Rysunek 11

Piła (kształ ten można też nazwać batarang – od broni Batmana) powstaje z pięciu półokręgów. Jednego dużego i czterech małych. Ważne w tym zadaniu jest też to, by panować nad kierunkiem, w którym będzie podążał duszek.

Rysunek 12: Kwiatek



Podpowiedzi
Rysunek 12

Kwiatek rysujemy tak samo, jak wcześniej rysowaliśmy rozetę (rysunek 8). Różnica polega na tym, że obracamy okrąg wokół jednego punktu.

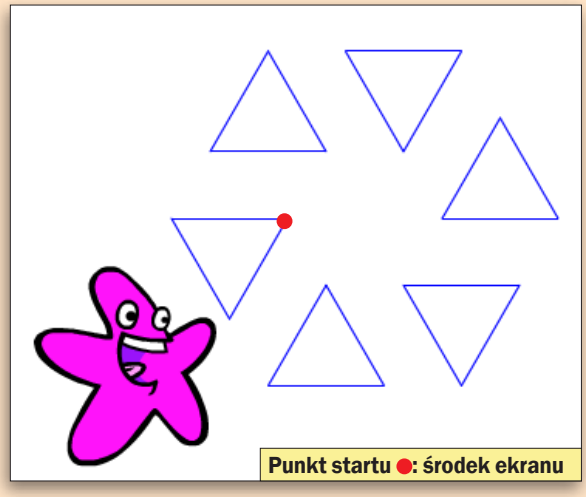
Rozwiązanie
Rysunek 11

Metodę rysowania półokręgu możemy wzorować na tym, jak wcześniej robiliśmy okrąg. Powtórzeń musi być tu jednak o połowę mniej, czyli 180. W dużym okręgu długość kroku duszka ustawimy na 2 kroki, a w małych na 0,5 kroku. Po narysowaniu każdego półokręgu dobrze też zastanowić się, jak duszek powinien pójść dalej. Pomocny może okazać się bloczek **ustaw kierunek na...** .

Rozwiązanie
Rysunek 12

Duszek przemierza trasę w kształcie okręgu, a po jej ukończeniu obraca się o kąt 10 stopni. Gdy takich powtórzeń ustalimy 36, to nasz program narysuje kwiatek.

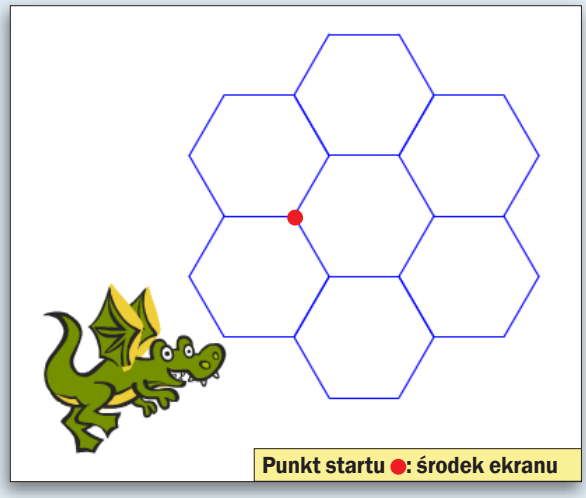
Rysunek 13: Trójkąty



Podpowiedzi
Rysunek 13

Zauważmy, że gdy połączymy wierzchołki trójkątów, które są bliżej centrum figury, to powstanie sześciokąt foremny.

Rysunek 14: Plaster miodu



Podpowiedzi
Rysunek 14

Ten rysunek powstaje w analogiczny sposób jak poprzedni. Różnica polega na tym, że zamiast trójkątów będziemy rysować sześciokąty.

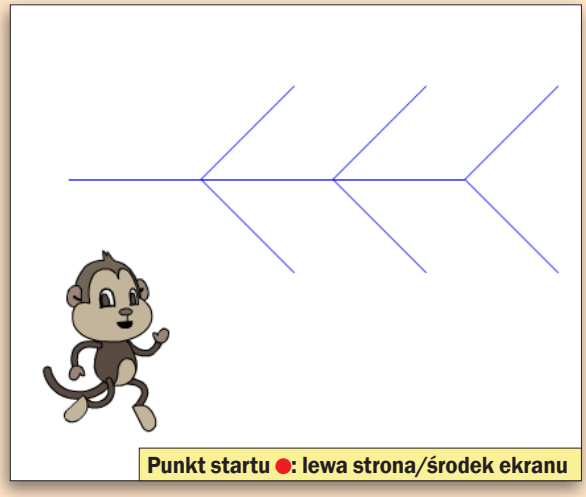
Rozwiązanie
Rysunek 13

Dla zwiększenia czytelności kodu skrypt rysujący trójkąt został wyeksponowany jako osobny bloczek. Sam rysunek powstaje przez przesunięcie duszka o pewną odległość bez zaznaczania śladu za nim, a następnie przez narysowanie trójkąta. Duszek potem wraca do centralnego punktu ekranu i obraca się o kąt 60 stopni.

Rozwiązanie
Rysunek 14

Wszystkie czynności w tym skrypcie będziemy wykonywali tak, jak w poprzednim zadaniu. Zwróćmy jednak uwagę, że odległość, o jaką przesuwa się duszek od środka, musi być identyczna jak długość boku rysowanego sześciokąta.

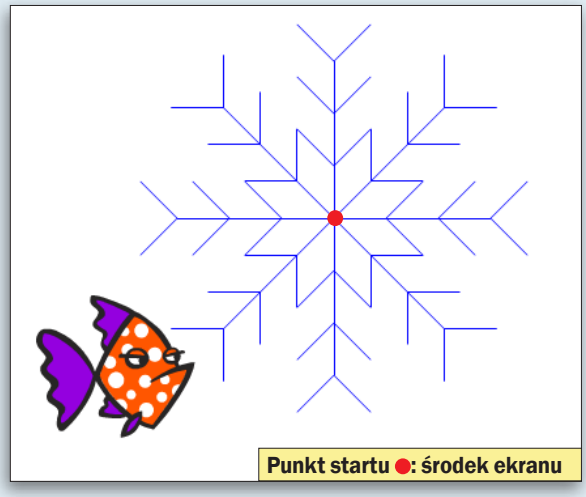
Rysunek 15: Strzałka



Podpowiedzi
Rysunek 15

Tę dziwną strzałkę stworzymy, trzykrotnie rysując znak **Y**. Zauważmy, że możemy nie tylko poruszać się o dodatnią liczbę kroków, ale też i o ujemną. Przykładowo, bloczek **przesuń o -5 kroków** sprawi, że nasz duszek przesunie się o 5 kroków do tyłu.

Rysunek 16: Śnieżynka



Podpowiedzi
Rysunek 16

Śnieżynka powstaje przez obracanie strzałki z poprzedniego ćwiczenia. By ułatwić sobie pracę z kodem, skrypt tworzący strzałkę stwórz jako osobny bloczek.

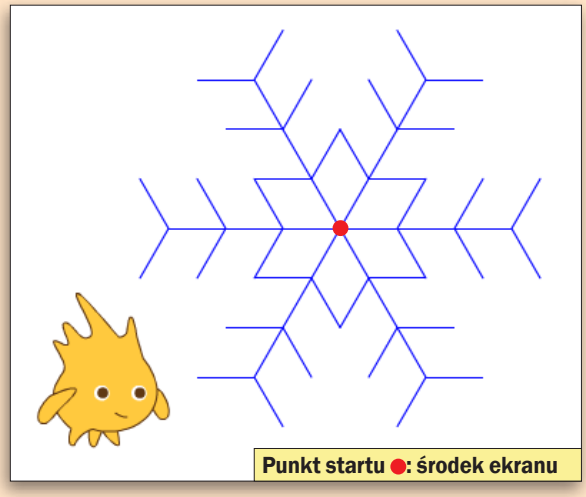
Rozwiązanie
Rysunek 15

Kod naszego programu składa się z trzykrotnego narysowania poziomej literki **Y**. Korzystamy tu z możliwości poruszania się nie tylko do przodu, ale też i do tyłu – poprzez wstawianie ujemnej liczby kroków. Dodając odpowiednie kąty obrotu, narysujemy naszą strzałkę.

Rozwiązanie
Rysunek 16

Gdy mamy już skrypt rysujący strzałki, całe zadanie jest proste. Program musi tylko narysować taką strzałkę, wrócić do punktu początkowego, a następnie obrócić się o kąt 45 stopni. Powstanie ośmioramienna śnieżynka.

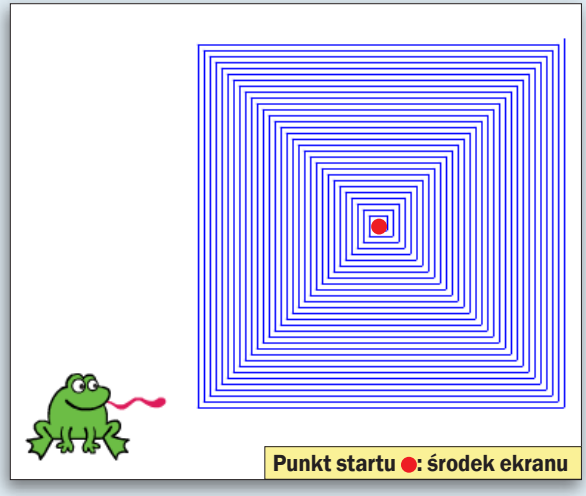
Rysunek 17: Śnieżynka 2



Podpowiedzi
Rysunek 17

Lekka modyfikacja wcześniejszego zadania, która sprawia, że nasza śnieżynka będzie miała sześć ramion. Wystarczy pobawić się kątami obrotu.

Rysunek 18: Labirynt



Podpowiedzi
Rysunek 18

Aby narysować labirynt, taki jak ten obok, musimy koniecznie zastosować zmienne. Rysunek, jak widać, powstaje bardzo podobnie jak kwadrat. Różnica polega na tym, że co przejście wydłuża się bok.

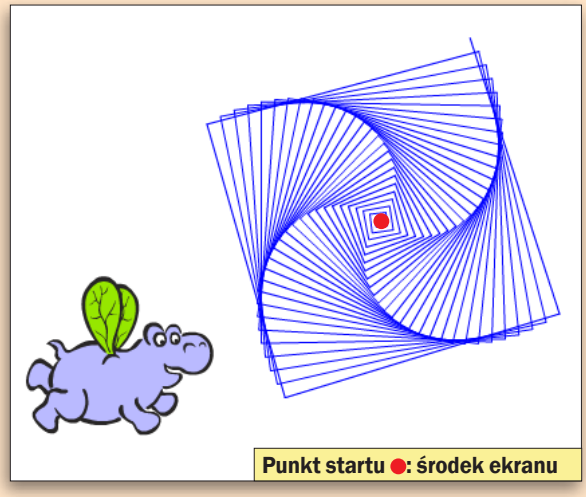
Rozwiązanie
Rysunek 17

Ta wersja śnieżynki wymaga kilkakrotnego użycia kąta 60 stopni (lub jego wielokrotności). Poza tym sam skrypt nie różni się działaniem od skryptu pierwszej śnieżynki.

Rozwiązanie
Rysunek 18

Kod programu niewiele różni się od tego służącego do tworzenia kwadratu. Co krok w naszej pętli będziemy jednak zwiększać liczbę kroków o 2 (wtedy powstaje 1 krok odstępu między równoległymi liniami). Program będzie działał do czasu, kiedy duszek dotknie krawędzi ekranu wykonywania aplikacji, co zapewniliśmy użyciem pętli z odpowiednim warunkiem.

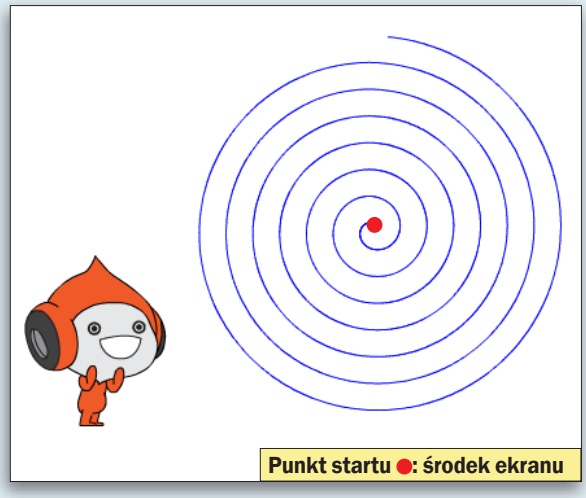
Rysunek 19: Galaktyka



Podpowiedzi
Rysunek 19

Galaktyka stworzona na rysunku obok to rozwinięcie pomysłu użytego w Labiryncie. W kodzie programu wystarczy dokonać jednej zmiany.

Rysunek 20: Spirala



Podpowiedzi
Rysunek 20

Kolejne rozwinięcie pomysłu z zadań Labiryt i Galaktyka. Wystarczy wybrać odpowiednią wartość, o jaką będzie się zmieniała liczba kroków przesunięcia duszka. Przy okazji, jeśli chcemy zapisać ułamek, powinniśmy użyć zielonego bloczka ... / ....

Rozwiązanie
Rysunek 19

Zmiana kąta obrotu z 90 stopni na 91 stopni powoduje, że rysowane kwadraty się zakrzywiają. Rysunek będzie się tworzył aż do momentu dotknięcia przez duszka krawędzi obszaru wykonywania programu. Ważne jest to, żeby zaczynać od wartości zmiennej równej 0. Dobrze jest również zawrzeć taki bloczek w naszej instrukcji.

Rozwiązanie
Rysunek 20

Jeżeli chodzi o główną część skryptu, to nie wymaga ona większego tłumaczenia. Wystarczy przejrzeć poprzednie zadania. Jeśli chodzi natomiast o to, o ile zmieniamy wartość zmiennej, to nie ma tu reguły. Im mniejsza zmiana, tym większe zagęszczenie naszej spirali.

ROZDZIAŁ 4

Proste programy

Pora zająć się nieco bardziej skomplikowanymi projektami. Z tego rozdziału dowiemy się, jak tworzyć proste programy – kalkulator czy tester refleksu

Nadeszła pora, byśmy zajęli się bardziej wymagającymi aplikacjami. Projekty, nad którymi wspólnie będziemy pracowali w tym rozdziale, pokażą, jakie są pełne możliwości środowiska programistycznego Scratch. Zajmiemy się tworzeniem takich programów, jak prosty kalkulator, tester refleksu czy konwerter tekstu na alfabet Morse'a. Najpierw stworzymy prosty program krok po kroku, a potem przećwiczymy zdobytą wiedzę na kolejnych zadaniach. Każdy projekt zaczniemy od obejrzenia efektu końcowego i powoli będziemy dążyć do jego osiągnięcia. Pamiętajmy, że metody realizacji tych projektów to tylko jedne z wielu różnych możliwości rozwiązania problemów. Można je ulepszać i modyfikować według swoich potrzeb. Skrypty ułożone według wskazówek przedstawionych w tym rozdziale znajdziemy na płycie dołączonej do książki.

DROGOWSKAZ

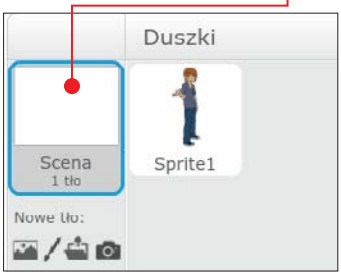
Jak tworzyć proste programy w Scratchu

Poprzednie rozdziały pozwoliły nam na opanowanie podstaw korzystania ze Scratcha, przećwiczyliśmy też nabyte umiejętności, tworząc rysunki. Teraz rozpoczniemy układanie bardziej rozbudowanych programów, które będą miały do spełnienia konkretne zadania. Poćwiczymy umiejętność pracy z instrukcjami warunkowymi i zmiennymi. Poznamy również działanie wielu nowych bloczków. Zaczniemy od dokładnego przeanalizowania projektu zawierającego niektóre z nowych dla nas bloczków. Nasz prosty program będzie losował liczbę od 1 do 100, a użytkownik będzie ją musiał zgadnąć, otrzymując od duszka podpowiedzi, czy jego typ jest liczbą za dużą czy za małą.

Zmieniamy kostium i tło sceny

1 W analogiczny sposób, jak to robiliśmy we wstępie do rozdziału **3**, zmieniamy kostium duszka. Wybierzmy na przykład duszka o nazwie **boy2-a**.

2 Teraz zmienimy tło sceny na jedno z dostępnych w bazie Scratcha. Klikamy na ikonę podglądu sceny.



3 U góry ekranu tekst **Kostiumy** zmienia się na napis **Tła**. Klikamy na niego. Zobaczymy okno bardzo podobne do tego, w którym zmienialiśmy kostium duszka.

4 Wybieramy ikonę przedstawiającą krajobraz podpisaną **Wybierz tło z biblioteki**. Otwiera się okno pozwalające na załadowanie nowego tła sceny. Wybieramy tło o nazwie **boardwalk**.

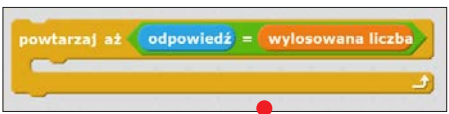
Kodujemy zachowanie duszka

1 By zacząć kodować, musimy najpierw sprawić, by program po kliknięciu na zieloną flagę wylosował liczbę od 1 do 100 i zapisał ją w zmiennej o nazwie **wylosowana liczba**. Trzeba taką zmienną stworzyć (patrz strona 23), a następnie wstawić do kodu bloczek **ustaw wylosowana liczba na...**. Wewnątrz pustego pola tekstowego wstawiamy klocek **losuj liczbę od 1 do 100**, oczywiście wpisując do niego odpowiednie wartości.

2 Jeśli uruchomimy teraz program, klikając na zieloną flagę, to w górnej części ekranu wykonywania aplikacji zo-

baczymy wylosowaną liczbę. Musimy to pole ukryć – gdyby było widoczne, użytkownik nie mógłby odgadywać liczby. Robimy to, włączając grupę bloczków **Dane**, a następnie usuwając zaznaczenie przy zmiennej **wylosowana liczba**.

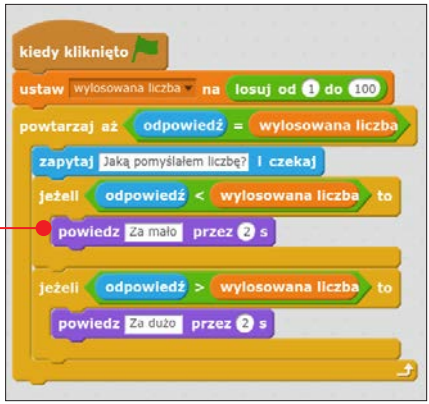
3 Teraz musimy sprawić, żeby program pytał użytkownika tak długo, aż otrzyma od niego w odpowiedzi właściwą liczbę. Oczywiście musimy do tego użyć pętli, która będzie działała, aż odpowiedź będzie równa wartości zmiennej **wylosowana liczba**. Wstawiamy zatem do kodu



pętlę **powtarzaj aż...** i zapełniamy jej puste pole warunkiem **odpowiedź = wylosowana liczba**.

4 Wewnątrz pętli powinno powtarzać się pytanie o liczbę, jaką typuje użytkownik. Tak jak w rozdziale **2** używamy do tego bloczka **zapytaj...**. Jako treść możemy wpisać na przykład tekst: **Jaką pomyślałem liczbę?**

5 Użytkownik odpowie, a więc tymczasowa zmienna **odpowiedź** będzie przechowywała tę liczbę. Możemy na niej popracować. Wstawmy do kodu dwie oddzielne instrukcje warunkowe



(patrz rozdział **2**), które będą powodowały, że jeśli **odpowiedź** będzie mniejsza od wartości zmiennej **wylosowana liczba**, to postać powie **Za mało**. Analogicznie, jeśli odpowiedź będzie większa, postać powinna powiedzieć **Za dużo**.

6 Najważniejsza część programu już działa. Teraz jeszcze ostatnia rzecz – poinformowanie użytkownika, że udało

Pełny kod programu

mu się odgadnąć liczbę. Zauważmy, że pętla będzie działała aż do momentu, gdy warunek równości **odpowiedzi** i **wylosowanej liczby** będzie spełniony. Tak więc w momencie, gdy użytkownik wpisze dobrą odpowiedź, pętla zakończy swoje działanie i rozpocznie się dalsza część kodu. To jest właśnie miejsce na wstawienie gratulacji dla użytkownika – zaraz za pętlą. Wstawmy tam bloczek **powiedz... przez 2 s** z tekstem **Udało się!!!**.

Ekran wykonywania aplikacji. Program wylosował liczbę od 1 do 100 i prosi użytkownika, by ją odgadł

Program 1: Prosty kalkulator

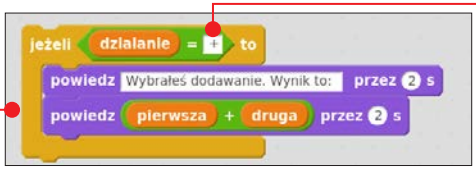
Celem tego projektu jest stworzenie prostego kalkulatora, w którym użytkownik będzie podawał dwie liczby oraz działanie, które chce wykonać. Aplikacja będzie zwracała wynik tej operacji. Użytkownik będzie mógł wybrać dodawanie, odejmowanie, mnożenie i dzielenie. Tworząc program, będziemy musieli pamiętać o zawarciu w nim obsługi błędów – aplikacja musi reagować, gdy użytkownik będzie chciał dzielić przez 0.

1 Zacznijmy od odpowiedniej oprawy graficznej programu. Zmieńmy duszka i tło sceny. Zamiast postaci kota możemy użyć na przykład niedźwiedzia polarnego, a jako tło sceny ustawić salę lekcyjną. Programy, które będziemy tworzyli, powinny być dopracowane pod każdym względem – również w aspekcie graficznym.

2 Nasz program musi zbierać w pierwszej kolejności dane w postaci dwóch liczb podanych przez użytkownika oraz działania, które ma zostać wykonane. Wykorzystamy w tym celu bloczki **zapytaj... i czekaj** oraz **odpowiedź**. Musimy też stworzyć zmienne (patrz strona 23), któ-

re będą pamiętały wybory użytkownika – dwie na liczby i jedną na działanie. Następnie układamy odpowiedni kod, który będzie pytał użytkownika, odczytywał jego odpowiedzi i na końcu zapisze je do odpowiednich zmiennych.

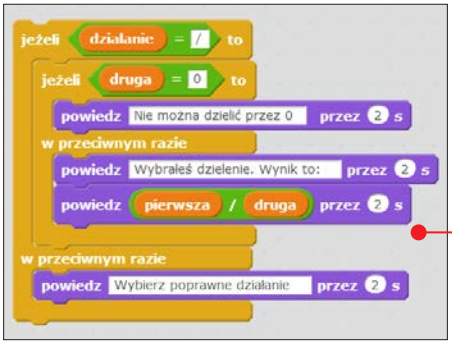
3 Już w tym momencie warto przetestować program, czy poprawnie zbiera dane. Kiedy uruchomimy go przez kliknięcie na zieloną flagę, a następnie odpowiemy na pytania naszego niedźwiedzia, u góry ekranu wykonywania aplikacji powinny pojawić się dane. Gdy tak się stanie, widoczne na rysunku pola możemy łatwo wyłączyć. Stosowne pola znajdziemy w grupie bloczków o nazwie **Dane**. Usuwając zaznaczenia przy nazwach ● zmiennych, sprawimy, że znikną one z ekranu aplikacji.



4 Przyszedł teraz czas na użycie instrukcji warunkowych **jeżeli...**. Sprawdzają one warunek postawiony przez programistę w pustym polu. Jeśli jest on prawdziwy – wtedy instrukcja jest wykonywana, jeżeli tak nie jest, to program ją pomija i idzie do dalszej części skryptu. Spróbujmy stworzyć instrukcję, która rozpozna, że użytkownik wpisał znak dodawania, a następnie poda w odpowiedzi wynik. Powinna ona wyglądać podobnie jak kod ●. Oczywiście dołączamy go do wcześniejszej instrukcji odczytującej dane.

Instrukcja sprawdza, czy zmienna **działanie** jest równa znakowi ●. Jeśli tak będzie, czyli jeśli użytkownik wpisze w trzecim pytaniu znak dodawania, to wartość logiczna tej instrukcji będzie prawdziwa.

Skrypt przejdzie wtedy do wykonywania fragmentu wewnątrz instrukcji warunkowej. Tam program ma za zadanie poinformować użytkownika, że wybrał dodawanie, a następnie powiedzieć, ile wynosi wartość wyrażenia **pierwsza + druga**, czyli zwrócić sumę liczb.



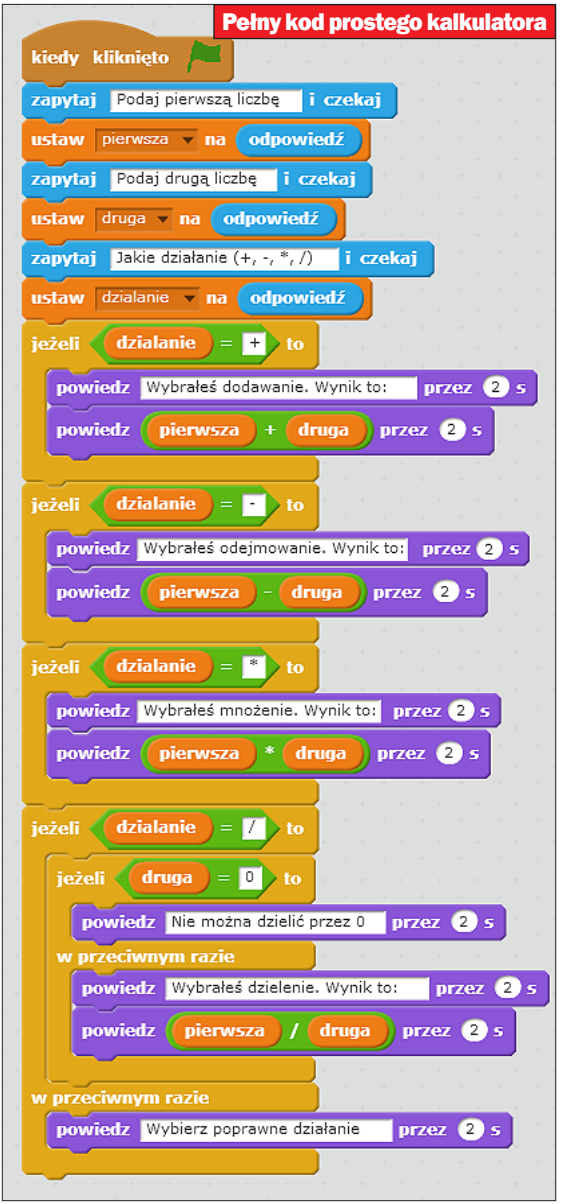
5 W analogiczny sposób będziemy postępowali, dodając instrukcję warunkową do odejmowania i mnożenia. W tym wypadku będzie się zmieniał tylko jeden bloczek odpowiedzialny za działanie wykonywane w odpowiedzi.

6 W dzieleniu nie wstawimy już bloczka **jeżeli...**, ale **jeżeli... w przeciwnym razie...**. Wytłumaczenie jest proste – gdy użytkownik poda inny niż wymagany przez nas znak działania – zostanie poinformowany o tym, że taka operacja nie jest możliwa i musi zacząć od początku.

7 Nie możemy jednak pozostawić dzielenia takim, jakie jest. Gdyby użytkownik wpisał jako drugą liczbę 0, to program nie wyliczyłby żadnego wyniku. Przez 0 nie można przecież dzielić. Musimy w takim razie odpowiednio zabezpieczyć poprzez wstawienie wewnątrz instrukcji warunkowej dzielenia kolejnej, która będzie sprawdzała, czy druga liczba jest zerem. Dopiero wtedy, gdy będziemy pewni, że tak nie jest, możemy wykonać dzielenie i podać wynik użytkownikowi.

8 Nasz kalkulator jest już gotowy. Nie jest oczywiście idealny. Wystarczy, że użytkownik zamiast liczb poda ciągi liter i program nie zadziała. Ominięcie tego błędu wymagałoby stworzenia bardzo rozbudowanej instrukcji warunkowej.

Pełny kod prostego kalkulatora



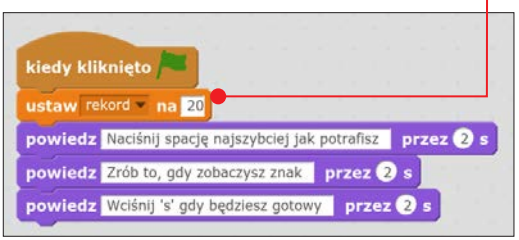
Program 2: Tester refleksu

Ten projekt pozwoli nam na stworzenie miernika czasu reakcji. Będzie on zapisywał najlepsze rezultaty użytkowników w danej sesji. Tworząc ten program, nauczymy się odczytywać wciśnięcie klawisza klawiatury. Poznamy również metodę losowania liczby z danego przedziału, co sprawi, że nasz program będzie nieprzewidywalny dla użytkowników.

1 Zacznijmy od odpowiedniej oprawy graficznej. Ustawmy duszka i tło, które według nas będą najbardziej odpowiednie.

2 Potem dodajmy trzy nowe zmienne o nazwach **losowy czas**, **wynik** i **rekord**. Pierwsze dwie ukryjmy tak, by w oknie aplikacji była widoczna tylko wartość zmiennej **rekord**. Dzięki temu użytkownicy cały czas będą widzieli najlepszy rezultat w danej sesji.

3 Po uruchomieniu programu sprawimy, by pojawiły się instrukcje dotyczące jego użytkowania. Muszą być krótkie i zawierać niezbędną treść potrzebną do zrozumienia jego zasady. Powinny informować o tym, że po stosownym sygnale użytkownik będzie musiał jak najszybciej wcisnąć spacja, by zatrzymać licznik czasu. Dodatkowo, podczas pierwszego uruchomienia programu będziemy ustawiali wartość zmiennej **rekord** na 20 (sekund). Gdybyśmy nie określi początkowego rekordu, zostały on automatycznie ustawiony na 0 sekund i nikomu nie udałoby się szybciej wcisnąć spacja.



4 Po tej instrukcji zajmiemy się drugą częścią kodu, która będzie inaugurowana wciśnięciem klawisza S. Musimy więc zacząć od umieszczenia bloczka **kiedy klawisz s naciśnięty**. Zaraz po nim wstawiamy klocek programujący naszego duszka, by ostrzegał użytkownika, że za chwilę zacznie się mierzenie jego czasu.

UWAGA!

Zmienne możemy ukryć tak, by nie wyświetlały się na ekranie wykonywania aplikacji. Wystarczy, że w pomarańczowej grupie bloczków **Zmienne** usuniemy zaznaczenie przy tych, których nie chcemy pokazywać użytkownikowi.

5 Teraz przyszła pora na wstawienie instrukcji, która sprawi, że nasz program odczeka losową liczbę sekund, a następnie pokaże sygnał do naciśnięcia spacji. Użyjemy do tego celu zmiennej **losowy czas**, bloczka **losuj od... do...** oraz bloczka wyliczającego iloraz dwóch liczb. Ustawmy wartość zmiennej na losową wartość od 1 do 50, podzieloną przez 10. Dlaczego? Sam bloczek odpowiedzialny za losowanie zwracałby tylko liczby 1, 2, 3, 4 itd. Dałoby się więc przewidzieć, kiedy może pojawić się sygnał. My, dzieląc liczby całkowite od 1 do 50 przez 10, otrzymamy czasy od 0,1 do 5 sekund z krokiem równym jednej dziesiątej sekundy. To o wiele

trudniejsze do przewidzenia. Po ustaleniu czasu losowania za pomocą bloczka **czekaj... s** sprawimy, że program będzie przez pewien czas bezczynny. Po tym czasie powinien dać znak użytkownikowi, na przykład powiedzieć **Teraz!!!**.

6 Kolejny fragment kodu będzie poświęcony mierzeniu czasu użytkownika, a następnie podaniu mu go przez duszka. W jasnoniebieskiej grupie bloczków o nazwie **Czujniki** znajdziemy bloczki **zeruj stoper** i **stoper**. Pierwszy sprawia, że licznik czasowy, uruchomiony wraz ze startem programu, zostanie zresetowany, czyli otrzyma wartość 0. Od razu potem zacznie naliczać czas od początku. Drugi z nich przechowuje aktualną wartość stopera. Co musi zrobić nasz program? Wyzerować w pierwszej kolejności stoper, a następnie czekać, aż użytkownik wciśnie spacja. Wtedy zmienna **wynik**

powinna być przyrównana do wartości bloczka **stoper**. Duszek powinien również powiedzieć, ile wyniosła ta wartość, przedstawiając wartość zmiennej **wynik**.

7 Ostatnia część programu będzie obsługiwała najwyższy wynik w danej sesji gry. Musimy wstawić porównanie wartości rekordu i obecnego wyniku użytkownika. Jeśli wynik będzie mniejszą liczbą sekund, to ma on stać się nowym rekordem – musi wte-

UWAGA!

Bloczek losuj **od... do...** sprawi, że w jego miejsce, w momencie wykonywania programu, zostanie wstawiona liczba całkowita z przedziału, który wewnątrz niego wpiszemy. Za każdym uruchomieniem programu ta liczba może być inna.

dy więc nastąpić nadpisanie zmiennej **rekord** nową wartością. Pomysł instrukcji warunkowej nasuwa się sam. Pod koniec programu trzeba poinstruować użytkownika, że gdy będzie chciał zrobić kolejne podejście do naszego refleksomierza, wystarczy, że wciśnie literę S.

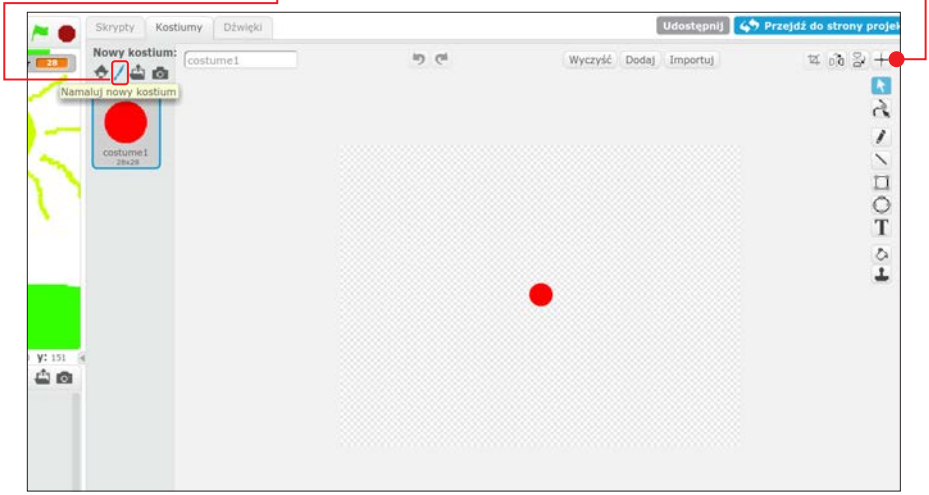
8 Nasza aplikacja jest gotowa. Możemy się nią pochwalić przed innymi i zająć się biciem kolejnych rekordów. Aby poćwiczyć, możemy spróbować wstawić inny sygnał do naszego miernika – dźwiękowy czy nawet wizualny. Duszek może zmieniać kolor albo wydać dźwięk klaksonu. Warto sprawdzić, czy ten zabieg sprawi, że wyniki się poprawią, czy może pogorszą.

Pełny kod testera refleksu

Program 3: Prosty program do malowania

Ten projekt to program do malowania – podobny do TuxPainta. Program nie będzie może miał profesjonalnych możliwości, ale jak na aplikację stworzoną z niewielkiej liczby bloczków – i tak będzie imponujący. Nasze dzieło będzie pozwalało na rysowanie dowolnie wybranym kolorem pędzla, którego rozmiar również będziemy mogli zmieniać. Dla uproszczenia obsługi zmiana tych parametrów będzie możliwa za pomocą klawiszy na klawiaturze.

1 Tworzenie zaczniemy od przygotowania duszka. Tym razem będzie to narzędzie – pędzel, którym będziemy malować. Przejdźmy do zakładki **Kostiumy** widocznej w górnej części ekranu. Klikając na ikonę pędzla, podpisaną jako **Namaluj nowy kostium**, stworzymy obrazek wyglądający jak czerwona kropka. Będzie to nasz pędzel. Teraz powinniśmy ustalić środek koła. Klikamy na ikonę w prawym górnym rogu okna **Kostiumy**, wyglądającą jak znak dodawania. Ustawiamy przecięcie linii poziomej i pio-



nowej dokładnie w środku pędzla. Na koniec możemy spokojnie usunąć niepotrzebne już kostiumy z kotem.

2 Pora na układanie kodu. Zacznijmy od stworzenia dwóch zmiennych o nazwach **kolor** i **rozmiar**. Będą one spełniały w naszym programie wyłącznie funkcję informacyjną dla użytkownika, więc zostawmy je widoczne na ekranie wykonywania aplikacji. Od razu możemy też zainicjować najważniejsze instrukcje, które będą wykonywane po kliknięciu na zieloną flagę. Aplikacja domyślnie będzie ustawiała rozmiar duszka na 100% oraz zapisywała go do zmiennej **rozmiar**. Duszek następnie zostanie ukryty – w przeciwnym wypadku rysowanie byłoby niemożliwe. Scratch cały czas myślałby, że klikając na duszka, tak naprawdę chcemy go przesunąć, a nie nim malować.

3 Teraz czas na wstawienie instrukcji, dzięki której nasz program zacznie rysować. Program powinien cały czas czuwać, wyczekując kliknięcia ze strony użytkownika. Kiedy ta czynność nastąpi, duszek powinien błyskawicznie przenieść się w miejsce kliknięcia i „odbić się tam". Użyjemy oczywiście pętli **zawsze**, w której wstawimy instrukcję warunkową sprawdzającą, czy nastąpiło kliknięcie myszą. Wewnątrz instrukcji

UWAGA!

Bloczki **x myszy** i **y myszy** z grupy **Czujniki** zwracają do programu aktualne współrzędne wskaźnika, o ile tylko znajduje się nad ekranem wykonywania aplikacji. Jeśli tylko ruszymy wskaźnikiem, wartości zawarte w każdym z bloczków też się zmienią.

jeżeli... to umieścimy bloczek pozwalający na natychmiastowe przeniesienie się w miejsce opisane współrzędnymi **x** i **y** – **idź do x:... y:...**. Bloczki zwracające aktualne współrzędne myszy znajdziemy w jasnoniebieskiej grupie **Czujniki**. Gdy duszek przejdzie już w odpowiednie miejsce, musi się tam odbić. Tę czynność zapewni nam bloczek **stempluj**. Po ułożeniu bloczków możemy przetestować program – powinien już rysować.

4 Dodamy teraz obsługę kilku klawiszy. Przypiszemy im działania polegające na zwiększaniu i zmniejszaniu pędzla oraz czyszczeniu ekranu. Oczywiście przyciski muszą działać niezależnie od działania programu – użytkownik może przecież w dowolnym momencie chcieć wykonać jedną z tych czynności. Użyjemy do tego bloczków **kiedy klawisz...**

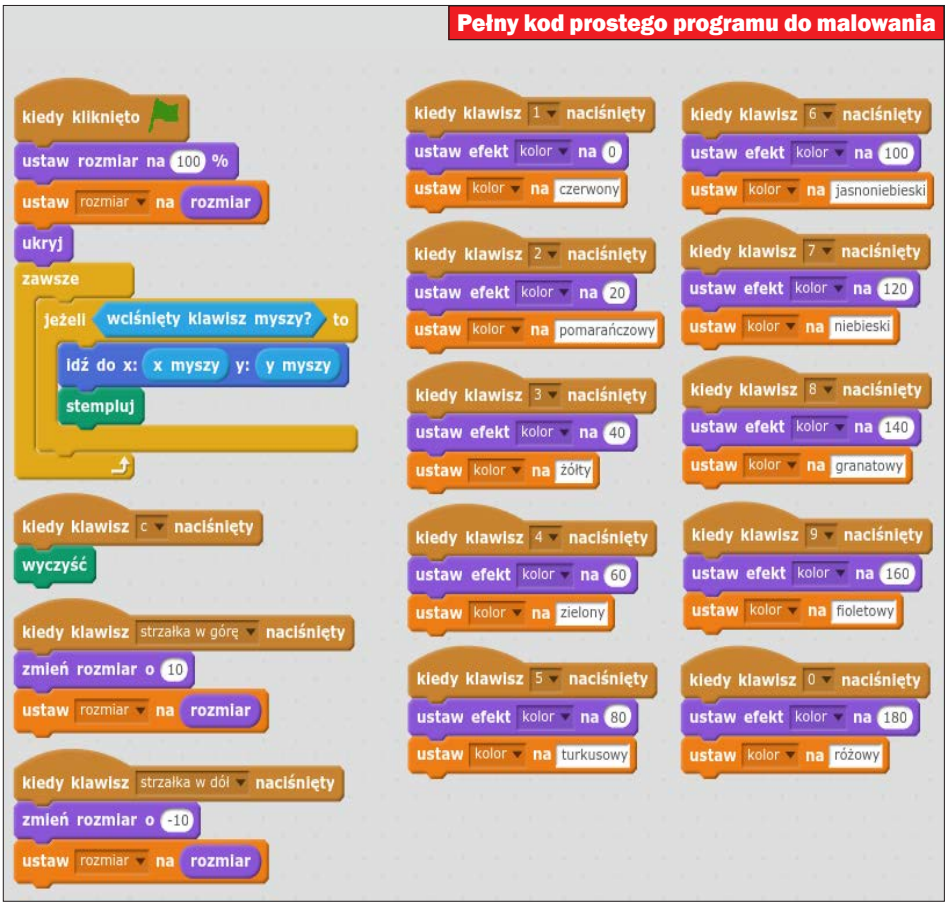
naciśnięty. Ustalimy czyszczenie ekranu na klawisz C, a obsługę wielkości pędzla na ↑ i ↓. Gdy już wszystko ułożymy, sprawdźmy, czy program działa poprawnie.

5 Teraz pora zająć się kolorami, którymi będziemy mogli malować. Ich zmiany oprzemy na liczbach od 0 do 9. Bloczek, który pozwoli nam zmieniać kolor pędzla, to **ustaw efekt kolor na...**. Musimy też sprawić, by po każdej zmianie koloru jego nazwa pojawiła się w zmiennej kolor. Użytkownik będzie wtedy wiedział,

jaką barwę ma do dyspozycji bez potrzeby jej próbowania. Nasz program jest już skończony. Możemy zacząć tworzyć rysunki.

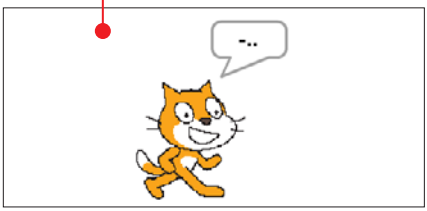
6 Aplikację możemy spróbować rozwinąć. Dobrym ćwiczeniem będzie dodanie do palety kolorów możliwości malowania na czarno i biało. Inną kwestią są kształty pędzla czy też umożliwienie użytkownikowi wyboru koloru tła wraz z uruchamianiem programu. Te opcje nie będą trudne do dodania, warto więc poświęcić chwilę na dopracowanie programu w ten sposób, a nawet na wymyślenie własnych dodatków.

Pełny kod prostego programu do malowania

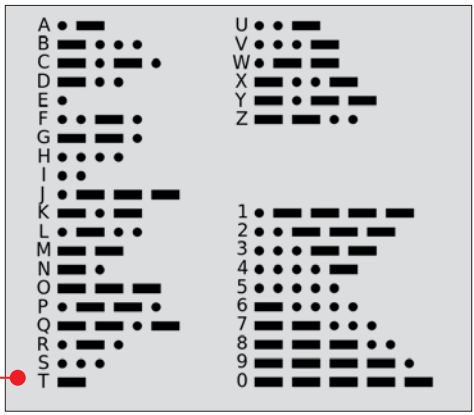


Program 4: Konwerter tekstu na Morse'a

Kolejna aplikacja, którą się zajmiemy, będzie tłumaczyła słowa na alfabet Morse'a. Powinna: odbierać wyraz



wprowadzony przez użytkownika, dzielić go na litery, zamieniać je na znaki alfabetu Morse'a i podawać użytkownikowi. Jak widzimy, metoda działania jest banalnie prosta. Teraz tylko musimy odpowiednio ułożyć bloczki. Bardzo ważne okażą się bloczki z grupy **Wyrażenia** – **długość...** oraz **litera.... z...**.



1 Zacznijmy od dodania zmiennej **litera**, która będzie bardzo krótko przechowywała w sobie poszczególne litery wyrazu podanego przez użytkownika. Wraz z inicjacją programu będzie musiała również zostać ustawiona na wartość równą 1. Po pierwsze, dlatego że program musi działać po każdym naciśnięciu zielonej flagi. Gdybyśmy nie ustawili wartości zmiennej przy starcie, kolejne użycie wymagałoby odświeżenia strony. Po drugie, wartość 1 jest ustawiona dlatego, że nasz program ma zamieniać litery, począwszy od pierwszej, a nie od domyślnie ustawionego zera.

2 Teraz zajmiemy się najważniejszą częścią aplikacji. Będzie ona zajmowała się zamianą liter wyrazu na odpowiadające im zapisy w alfabecie Morse'a. Musimy wstawić tam pętlę, która będzie miała tyle powtórzeń, ile podane przez użytkownika słowo będzie miało znaków. Uzyskamy to dzięki bloczkowi **długość...** ze wstawionym w niego klockiem **odpowiedź**.

Dalej, korzystając z bloczka **nadaj...**, wybieramy literę o takim numerze, jaki obecnie ma nasza zmienna. Skrypt, który stworzymy za chwilę, będzie odpowiadał na to zawołanie. Do pętli musimy jeszcze dodać bloczek zmieniający zmienną **liczba** o 1 tak, by po następnym przejściu zamieniana była następna z liter.

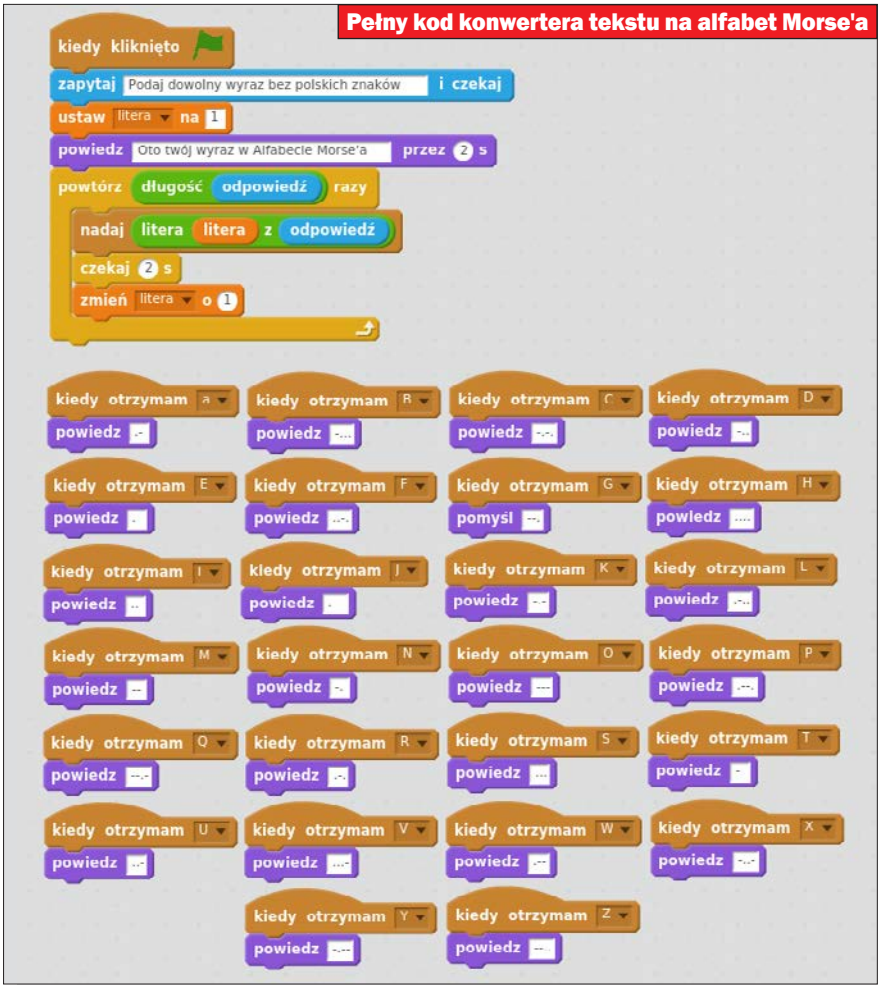
3 Kolejne bloczki, które dodamy, będą odpowiedziami na to, co program będzie nadawał w trakcie wykonywania pętli. Będzie to wiele niezależnych skryptów postaci **kiedy otrzymam** a, **kiedy otrzymam** b ● i tak dalej. Każdy z nich będzie mógł zostać wywołany, gdy tylko zostanie o to „poproszony" przez program. Wewnątrz każdej z tych instrukcji wstawiamy tylko bloczek **powiedz...** z tłumaczeniem wywoływanej litery. Takich skryptów musimy wstawić tyle, ile jest liter w alfabecie.

UWAGA!

Klocek **długość...** zwraca liczbę liter wyrazu wpisanego w pole tekstowe bloczka. Jeśli w programie wstawilibyśmy klocek **długość kot**, to aplikacja traktowałaby go jako liczbę 3.

UWAGA!

Bloczek **litera... z...**, jeśli zostanie wykonany w programie, to da nam w odpowiedzi literę ustawioną na określonym miejscu w wyrazie. Przykładowo bloczek **litera 2** z wyrazu kot da nam literę o.



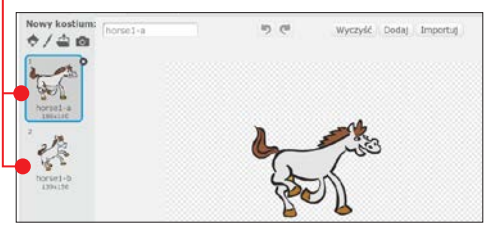
4 Możemy uruchomić program i sprawdzić, czy wszystko działa poprawnie. Po wpisaniu dowolnego słowa po kolei powinny się wyświetlać tłumaczenia poszczególnych liter. Możemy go teraz ulepszać. Będzie robił o wiele większe wrażenie, jeśli na przykład do każdej litery, oprócz tekstu, dodamy efekt dźwiękowy albo nawet świetlny działający jako kreska lub kropka. To nie jest trudna czynność, szczególnie że trzon programu mamy już przecież ukończony.

Program 5: Wyścigi

Ten projekt będzie polegał na stworzeniu programu symulującego wyścig dwóch koni. Wyścig będzie o tyle ciekawy, że o wygranej w nim zdecyduje los – każde ze zwierząt będzie poruszało się o losową liczbę kroków do przodu. Postaramy się również tak zanimować ruch duszków, by rzeczywiście wyglądały, jakby biegły. Gdy duszek zwycięzca dobiegnie do mety, program pokaże jego imię w polu zmiennej nad torem wyścigowym. W działaniu program będzie wykorzystywał jedną zmienną (patrz strona 23) o nazwie **zwycięzca** przechowującą nazwę konia, który wygrał ostatni wyścig.

1 Zacznijmy od zapewnienia programowi odpowiedniej oprawy graficznej. Stwórzmy dwa duszki, które nazwiemy na przykład **Koń nr 1** i **Koń nr 2**. W zakładce **Kostiumy**, korzystając z dostępnych w bazie Scratcha grafik, wybierzmy dwa kostiumy konia (znajdują się one w grupie **Transport**) dla jednego duszka ● i dwa dla drugiego. Korzystając z edytora (narzędzia wypełniania), zmieńmy również kolor jednego ze zwierząt.

2 Teraz tło sceny – stworzymy je samodzielnie. Nasze tło będzie składało się z dwóch torów będących prostokątami, trawy oraz skrawka nieba ●. Na torach wyścigowych umieśćmy cyfry, pozwalające zidentyfikować konie.



3 Przyszedł czas, by zaprogramować pierwszego konia tak, aby z losową szybkością przemierzał długość toru. Zacznijmy od ustawienia duszka. Przenieśmy go myszą na początek toru ● tak, by jego ogon nie dotykał krawędzi ekranu wykonywania aplikacji. To ważne, bo nasz program będzie kończył wyścig, gdy koń dobiegnie do końca toru – czyli właśnie dotknie krawędzi. Złe ustawienie spowodowałoby zakończenie działania aplikacji, zanim tak naprawdę zaczęłaby działać.

4 Skrypt konia musi działać tak, by duszek po kliknięciu na zieloną flagę zawsze ustawiał się na początku toru, a potem rozpoczynał pętlę trwającą do momentu najechania na krawędź ekranu. Każde przejście pętli ma polegać na przesunięciu się o losową liczbę kroków – od 0 do 2 ●. Koń będzie mógł, w zależności od szczęścia, przesunąć się o 0, 1 lub 2 kroki do przodu. Gdy tylko pętla skończy działanie, stworzona wcześniej zmienna **zwycięzca** powinna ustawić swoją wartość na tekst **Koń nr 1** i zakończyć działanie innych skryptów (czyli i wyścig drugiego konia).

Pełny kod wyścigów – Koń nr 1

5 Pierwszy koń powinien już poprawnie przemierzać trasę. Teraz dodamy animację ruchu. Skrypt ten będzie niezależny od poprzedniego, możemy go zbudować obok. Wykorzystamy dwa kostiumy, które są przydzielone do tego duszka. Animacja ma się zaczynać wraz ze startem programu i polegać na niekończących się zmianach kostiumu. Między tymi zmianami musimy umieścić krótką przerwę (inaczej wszystko działoby się tak szybko, że nie można by było zaobserwować zmian). Nie musimy się martwić o to, kiedy bieg konia ma się skończyć – załatwi to za nas poprzedni skrypt.

6 Skrypty regulujące działanie drugiego konia nie będą się różnić od skryptów tego pierwszego. (**Uwaga!** Scratch

Pełny kod wyścigów – scena

zmienia zawartość ekranu skryptów w zależności od aktywnego elementu – patrz ramka na stronie 62). Zamiast układać cały kod od nowa, możemy po prostu przeciągnąć skrypt z jednego duszka na drugiego. Wystarczy chwycić myszą bloczki jednej instrukcji i najechać nimi na drugiego duszka, widocznego na ekranie duszków. Zmianom ulegną współrzędne, na których po kliknięciu na zieloną flagę ma lądować duszek. Warto zauważyć, że żaden z duszków nie może w momencie startu znajdować się przed drugim – dlatego dokładnie sprawdźmy, czy współrzędna **x** jest u obu duszków taka sama.

7 Oba konie powinny się już ścigać po kliknięciu na zieloną flagę. Jeszcze tylko drobna poprawka. Po kliknięciu na zieloną flagę zmienna **zwycięzca**, mimo trwania wyścigu, informuje, że jeden z duszków jest zwycięzcą. Zmienimy to. Kliknijmy na ikonę sceny, widoczną pod ekranem wykonywania aplikacji, i wstawmy skrypt, który po rozpoczęciu działania aplikacji ustawi zmienną na tekst **Wyścig w trakcie**.

8 Aplikacja jest już skończona. Możemy ją ulepszyć. Spróbujmy dodać do niej efekty dźwiękowe, trzeciego konia, licznik wygranych poszczególnych zwierząt.

Pełny kod wyścigów – Koń nr 2

ROZDZIAŁ 5

Jak tworzyć gry

Kto nie chciałby być twórcą gier? Scratch pozwoli nam zobaczyć, na czym polega programowanie prostych aplikacji służących do zabawy

Tworzenie programów użytkowych to bardzo ciekawe zajęcie. Jednak jeśli chcemy zobaczyć pełen potencjał Scratcha, powinniśmy spróbować tworzyć gry. Tu będzie ważny nie tylko sam algorytm, ale też obudowa graficzna i dźwiękowa. W tym rozdziale stworzymy najpierw naszą pierwszą grę – krok po kroku, by nauczyć się, jak to robić. A następnie przećwiczymy zdobytą wiedzę, układając jeszcze pięć gier o różnej tematyce i o różnej mechanice działania. Dzięki tym projektom poznamy metody wzajemnego oddziaływania między dwoma duszkami, a także postaramy się zasymulować sztuczną inteligencję przeciwnika. Skrypty stworzone według wskazówek znajdziemy na płycie.

DROGOWSKAZ

Uczymy się tworzyć gry

Umiemy już wystarczająco dobrze tworzyć skrypty, by przejść o krok dalej. W tym rozdziale zaczniemy więc wspólnie programować gry w Scratchu. Stworzymy jeszcze bardziej zaawansowane skrypty, obsługujące relacje między kilkoma duszkami, i skupimy się na graficznej stronie naszych dzieł. Nowe skrypty będą wymagały od nas wykorzystania kilku nowych bloczków.

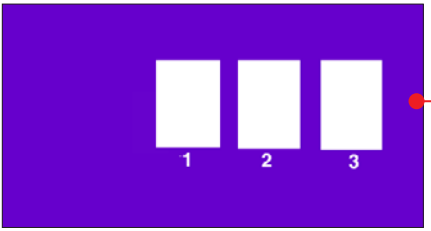
Pierwsza gra

Teraz postaramy się wspólnie stworzyć bardzo prostą grę, w której zadaniem gracza będzie odgadnięcie, za którą z trzech bramek schowało się jabłko. Poznamy obsługę punktacji dla użytkownika oraz samodzielnie stworzymy tło sceny. Zobaczymy też, w jaki sposób utworzyć interakcję między dwoma duszkami.

Dwa duszki i trzy warianty sceny

1 Tworzymy nowy projekt Scratcha. Zaczynamy od dodania dwóch duszków – jeden będzie małpką (**kostium monkey1-b**), a drugi jabłkiem (**kostium apple**). Niezbędne informacje, jak to zrobić, znajdują się w rozdziałach **3** i **4**.

2 Kolejnym krokiem jest stworzenie nowego tła dla sceny. Narysujemy je samodzielnie, korzystając z wbudowanego edytora grafiki. Przechodzimy do niego tak jak w rozdziale **4**. Teraz, używając narzędzi **Wypełnij kolorem**, **Prostokąt** i **Tekst**, tworzymy obraz jak poniżej. Każdy z białych prostokątów będzie bramką, za którą będzie się chowało jabłko.



3 Dla tej sceny stwórzmy jeszcze dwa warianty kolorystyczne. Wystarczy, że klikniemy na jej miniaturę i na przycisk **duplikuj**. Jeszcze raz używamy narzędzia **Wypełnij kolorem** i mamy trzy różne wersje naszej sceny.

Kodujemy

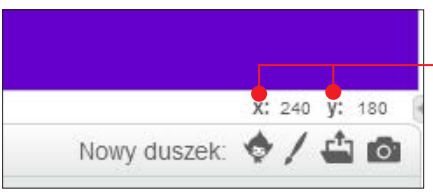
1 Przejdźmy do kodowania. Dodajmy do naszego programu zmienną o nazwie **wynik** (patrz rozdział **2**), to w niej będzie przechowywana liczba dobrych odpowiedzi udzielonych przez użytkownika.

2 Zacznijmy od małpki. Kiedy użytkownik kliknie na zieloną flagę, wszystko, co znajduje się w zmiennej **wynik**, powinno zostać skasowane, by gra zliczała punkty użytkownika od zera. Powinna również pojawić się informacja, że użytkownik ma wcisnąć spacja. To właśnie ten klawisz będzie obsługiwał w naszej grze rozpoczynanie nowych prób.

3 Przełączmy się teraz na duszka – jabłko. Zauważymy na pewno, że wszystkie bloczki zniknęły. Nic się jednak nie stało. Jeśli przełączymy się z powrotem na małpkę, wstawiony wcześniej kod znowu się pojawi.

4 Małpka na początku mówi do użytkownika, by wcisnął spacja. A jabłko musi zareagować na wciśnięcie spacja. Zacznijmy skrypt od wstawienia bloczka **kiedy klawisz spacja wciśnięty** z grupy **Zdarzenia**. Następnie duszek powinien zniknąć, by użytkownik nie wiedział, gdzie się zaraz przemieści. Stanie się tak, gdy wstawimy klocek **ukryj** z kategorii **Wygląd**.

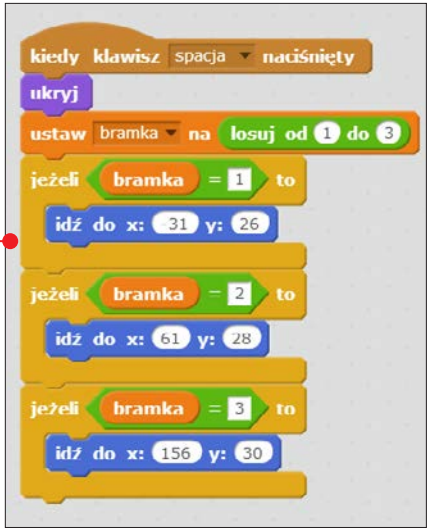
5 Zajmiemy się teraz wylosowaniem bramki i przemieszczeniem jabłka. Liczba oznaczająca wylosowaną bramkę musi być gdzieś przechowywana. Stwórzmy zatem kolejną zmienną o nazwie **bramka**. Jako ciąg dalszy skryptu jabłka dodajmy ustawienie wartości zmiennej **bramka** na **losową liczbę od 1 do 3**. Następnie wstawmy trzy instrukcje warunkowe, które w zależności od wylosowanej liczby umieszczą jabłko w jednej z trzech bramek. Współrzędne możemy odczytać, chociażby przesuwając w odpowiednie miejsce wskaźnik myszy.

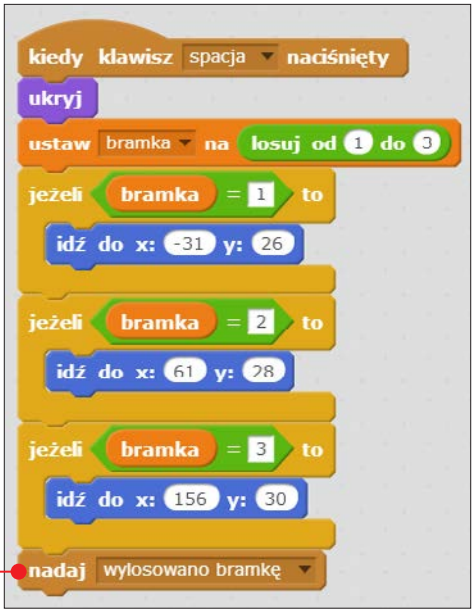


Zaraz pod ekranem wykonywania aplikacji odczytamy liczbę x i y. Wpiszmy

UWAGA!

Scratch zmienia zawartość ekranu skryptów w zależności od aktywnego elementu, którym może być jeden z duszków lub scena. Dla każdego z nich zobaczymy na ekranie inny kod. Nie da się zobaczyć wszystkich kodów naraz. Dlatego na końcu tej porady zostały zamieszczone trzy osobne ilustracje z pełnymi kodami – dwóch duszków i sceny.





je teraz po kolei w instrukcjach warunkowych.

6 Po ustawieniu jabłka we właściwej pozycji małpka musi zostać o tym powiadomiona. Uda nam się to zrobić, wstawiając za instrukcjami warunkowymi w kodzie jabłka bloczek **nadaj...** z komunikatem **wylosowano bramkę**.

7 Przejdźmy znowu do duszka – małpki. Musi on odebrać sygnał wysłany przez jabłko, a następnie spytać użytkownika, w której bramce jego zdaniem jest ukryte jabłko. Użyjemy bloczka **zapytaj... i czekaj**. Treść pytania powinna być tak sformułowana, żeby użytkownik wiedział, co wpisać. Może to być na przykład: **W której bramce jest jabłko (1,2,3)?**.

Pełny kod małpki

Pełny kod jabłka

8 Zaraz potem wstawimy instrukcję warunkową, która sprawdzi, czy odpowiedź użytkownika jest równa wartości zmiennej **bramka**. Jeśli tak jest, użytkownik powinien zobaczyć gratulacje, a zmienna **wynik** powinna być zwiększona o 1. Jeżeli użytkownik się pomylił, powinien zobaczyć komunikat, który zachęci go, by spróbował jeszcze raz.

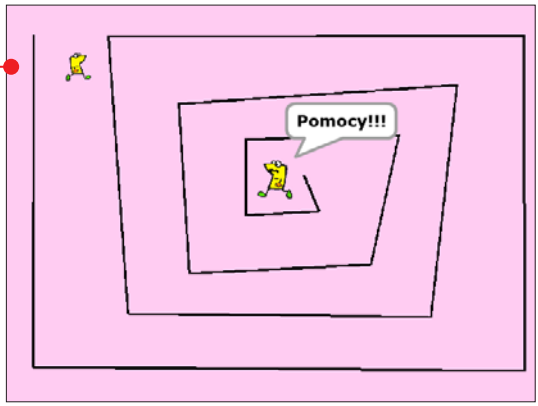
9 Dodajmy na końcu tego skryptu informację dla użytkownika o tym, że aby była możliwa następna próba, musi zostać wciśnięty klawisz spacja. Nadajmy również nowy sygnał, nazywając go **następna runda**.

10 Sygnał następnej rundy powinno otrzymać zarówno jabłko, jak i scena. W efekcie jabłko będzie pojawiało się po każdej, zarówno udanej, jak i nieudanej próbie, a tło sceny – będzie się zmieniało (zmianę sceny umożliwia bloczek **następne tło**).

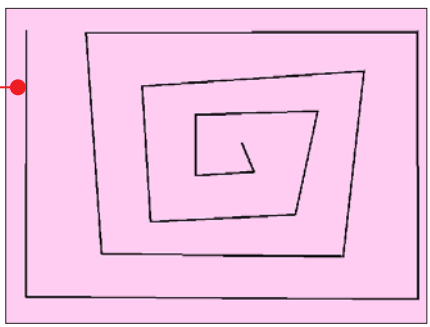
Pełny kod sceny

Gra 1: Duszek w labiryncie

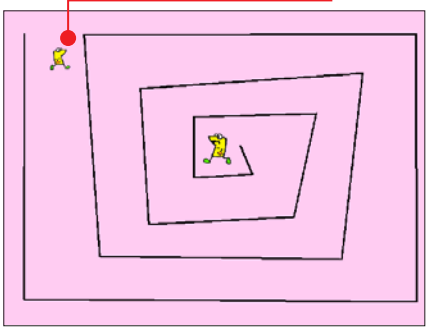
Gra, którą teraz stworzymy, będzie polegała na jak najszybszym przeprowadzeniu duszka przez labirynt do miejsca docelowego – u nas do wołającego o pomoc drugiego duszka. Bardzo podobne zabawki robiło się kiedyś z powyginanego kawałka metalu, gwoździa, żarówki i baterii. My ulepszymy ten pomysł, dodając do naszego programu stoper, który będzie zapisywał najszybszy czas przejścia. Programując aplikację, nauczymy się tworzyć interakcje między kursorem a postacią duszka, poćwiczymy również dodawanie interakcji między dwoma duszkami.

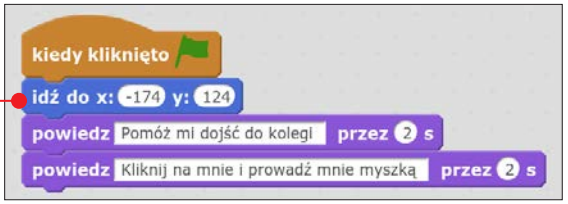


1 Zaczniemy od stworzenia odpowiedniego tła. Musimy na nim namalować labirynt, koniecznie czarną linią. Algorytm naszego programu będzie ustawiony na wykrywanie zetknięcia z czarnym kolorem. (Później, modyfikując naszą grę, jeśli zechcemy, będziemy mogli ustawiać dowolne inne kolory). Kształt labiryntu zależy od naszej inwencji. Ważne tylko, żeby szerokość drogi w labiryncie nie była zbyt mała, bo wtedy labirynt będzie niemożliwy do przejścia.



2 Teraz przygotujmy duszki. Potrzebne są dwie postaci. Nie ma wymagań co do ich wyglądu. Możemy je nawet samodzielnie namalować albo zaimportować z kamerki internetowej czy pliku w komputerze. Postarajmy się natomiast zmienić ich rozmiar na taki, by każdy z nich mógł spokojnie przejść cały labirynt. Gdy duszki zostaną już stworzone, ustawimy je w odpowiednich pozycjach. Jeden musi stać przy wejściu do labiryntu, a drugi w jego centrum.





3 Zaczynamy kodować. Popracujmy nad duszkiem, który będzie przechodził przez labirynt – dla porządku nazywajmy go duszkiem pierwszym. Po kliknięciu na zieloną flagę powinien ustawiać się zawsze w tym samym miejscu – zapewnimy to bloczkiem **idź do x:... y:...**. Następnie powinien wyświetlić instrukcje mówiące o tym, że użytkownik ma kliknąć na niego, a następnie poprowadzić przez labirynt aż do drugiego duszka.

4 Teraz program musi czekać aż do momentu, kiedy użytkownik kliknie na pierwszego duszka. Nie ma, niestety, gotowego bloczka, ale to nie problem. Korzystając z zielonego bloczka **... i...**, stworzymy taką instrukcję. Kliknięcie na duszka to tak naprawdę spełnienie dwóch warunków jednocześnie – naciskamy przycisk myszy i najeżdżamy kursorem na pierwszego duszka. Zaraz za tą instrukcją powinniśmy również wyzerować stoper. Teraz, żeby duszek „przylepił się" do kursora myszy, powinniśmy wstawić pętlę, która będzie działała, aż duszek dotknie czarnego koloru (czyli ściany labiryntu) albo drugiego duszka. W pętli musi być tylko jedna instrukcja – przejścia w aktualne współrzędne wskaźnika myszy.

5 Włączając program, widzimy już, że wszystko zaczyna działać poprawnie. Teraz przyszedł czas na wstawienie instrukcji warunkowej. Wiemy, że pętla trwa aż do momentu natrafienia na czarny bok labiryntu albo na drugiego duszka. Dlatego nasza instrukcja powinna sprawdzać, czy trafiliśmy na coś czarnego. Jeśli tak się stanie, gra się kończy. Jeśli nie, to oznacza, że gracz doszedł do drugiego duszka, więc powinien mu zostać podany czas, z jakim ukończył ćwiczenie.

6 Drugi duszek w środku labiryntu musi cały czas wołać o pomoc. Musi to robić od momentu wciśnięcia zielonej flagi aż do

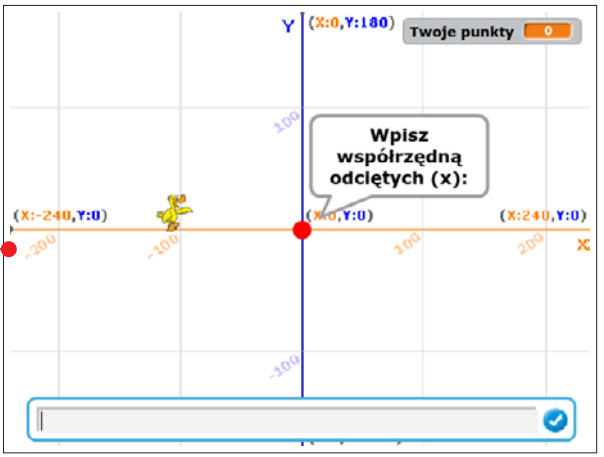
Pełny kod duszka w środku labiryntu

dotknięcia go przez pierwszego duszka. Gdy tylko postacie się spotkają, zostanie odtworzony dźwięk trąbki i drugi duszek bardzo się ucieszy.

7 Nasza pierwsza gra jest gotowa. W ramach dodatkowych ćwiczeń możemy zacząć ją ulepszać. Na przykład, po każdym pomyślnym przejściu labiryntu może on zmieniać się na inny – wystarczy dodać kilka scen i odpowiednio to oskryptować. Dobrym pomysłem jest również stworzenie tablicy wyników na 10 miejsc, gdzie każdy z zawodników mógłby zamieścić swój wynik opatrzony trzema wybranymi literami – tak jak to było w starych automatach z grami.

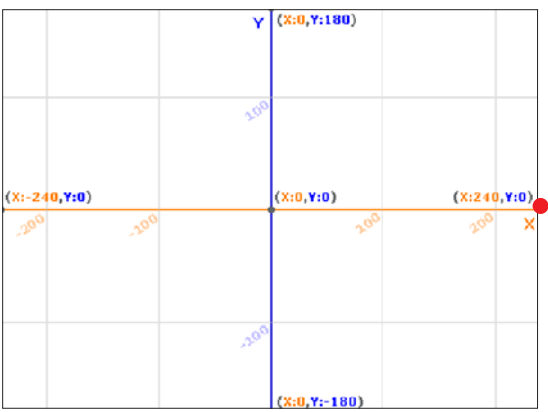
Pełny kod pierwszego z duszków

Gra 2: Polowanie na kaczki



Zajmiemy się teraz… polowaniem na kaczki. W naszej nowej grze zadanie gracza będzie polegało na odczytaniu położenia kaczki i wpisaniu jego współrzędnych tak, by najechał na nią czerwony celownik. Gra będzie trwała, dopóki użytkownik nie zdobędzie 10 punktów. Za każde trafienie otrzyma jeden punkt, a za pudło – tyle samo straci. Opracowując tę grę, lepiej zaznajomimy się z układem współrzędnych oraz poćwiczymy programowanie interakcji między dwoma duszkami.

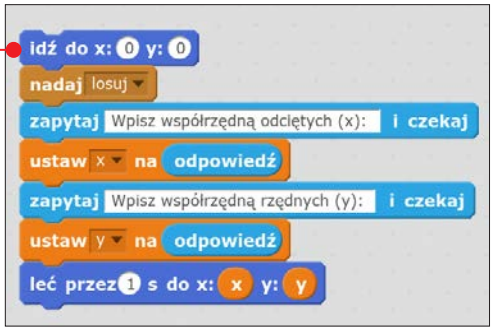
1 Rozpoczniemy od przygotowania odpowiedniego interfejsu graficznego. Potrzebne nam są dwa duszki – kaczka, którą znajdziemy w bazie kostiumów Scratcha, oraz czerwone kółko, które będzie celownikiem. Dodanego ptaka warto zmniejszyć, tak by trafienie w niego nie było zbyt łatwe. U góry ekranu znajdziemy potrzebne przyciski. Kolejnym elementem, potrzebnym, by w naszą grę przyjemnie się grało, jest odpowiednie tło. Takie, które pomagałoby w odczytywaniu współrzędnych, ale też nie sprawiało, że odpowiedzi będą podawane na tacy. W bazie obrazów dla scen, w kategorii **Inne**, znajdziemy tło o nazwie **xy-grid**.



2 Zacznijmy kodować. Na pierwszy ogień pójdzie czerwony celownik. Niech na początku, po kliknięciu na zieloną flagę, przesunie się na wierzch (żeby gdy najedziemy nim na kaczkę, nie znalazł się

pod nią), ustawi punktację na 0 oraz krótko opowie, jak działa program i jak wygląda punktacja. Potrzebujemy do tego stworzyć zmienną przechowującą liczbę punktów, którą będzie można wyświetlić użytkownikowi. Może nazywać się na przykład **Twoje punkty**.

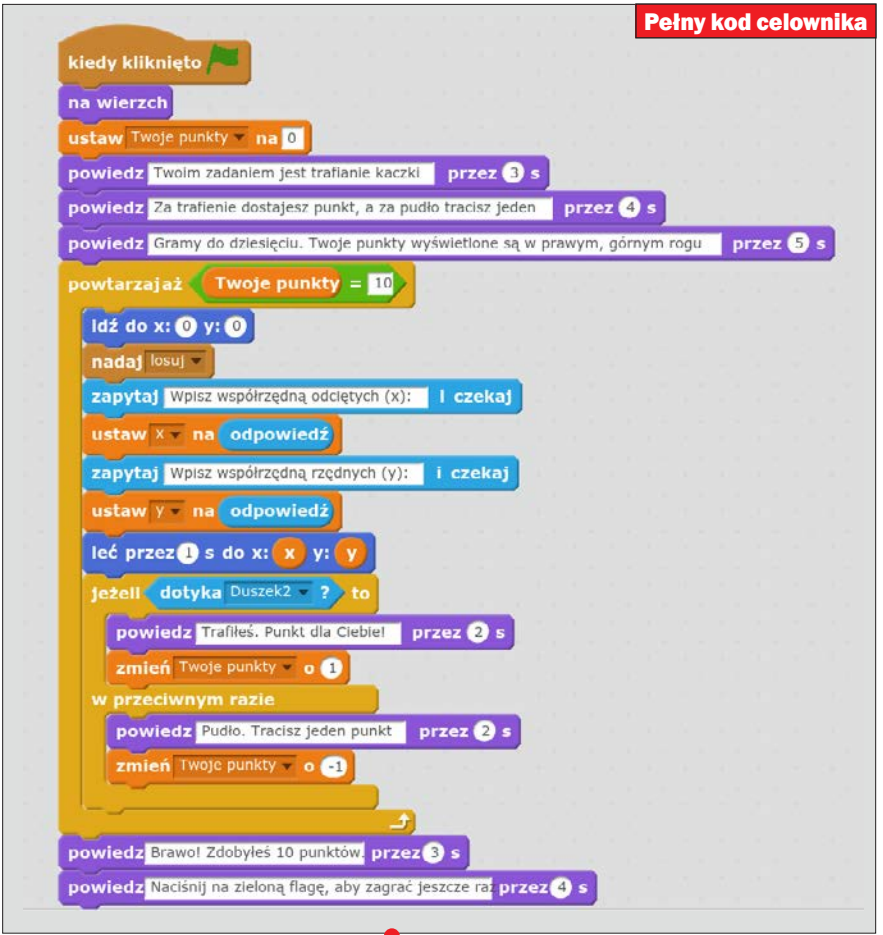
3 Teraz spróbujmy sprawić, by celownik powędrował w kierunku kaczki. Ustawimy duszka tak, że będzie startował od punktu (0, 0). Gdy to się stanie, nadamy sygnał dla kaczki, by wylosowała położenie na ekranie wykonywania programu – za chwilę przejdziemy do jej programowania. Potem trzeba będzie poprosić użytkownika o wpisanie współrzędnych kaczki. Te odpowiedzi zapiszemy w dwóch zmiennych. Może być to na przykład **x** i **y**. Następnie celownik powinien szybko przesunąć się we wskazane miejsce.



4 Dodajmy teraz obsługę dwóch stanów, w jakich może znaleźć się celownik po przejściu w miejsce, którego współrzędne podał gracz: albo trafił w kaczkę, albo nie. Jeśli tak, należy przyznać punkt, w przeciwnym wypadku go odjąć. Całą tę operację oskryptujemy jedną instrukcją warunkową, która będzie sprawdzała, czy drugi duszek został dotknięty.

5 Cała ta instrukcja będzie musiała być zawarta wewnątrz pętli, trwającej, dopóki liczba zdobytych punktów nie będzie równa 10. Po osiągnięciu tego celu należy oczywi-

Pełny kod celownika



ście pogratulować użytkownikowi. Oto gotowy kod dla celownika ●.

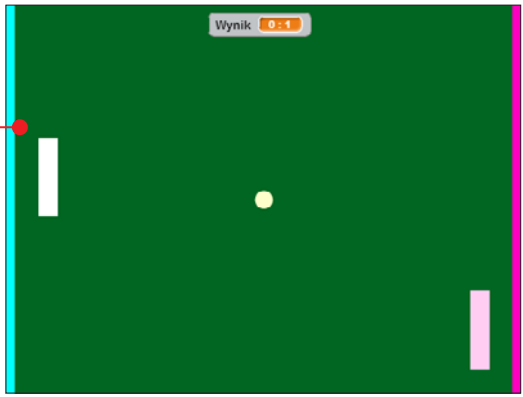
6 Zajmijmy się kaczką. Pamiętajmy, że tworząc skrypt dla celownika, wysyłaliśmy sygnał **losuj** ●. Teraz, kodując kaczkę, musimy ten sygnał przechwycić i sprawić, by po jego

Pełny kod kaczki

otrzymaniu przeniosła się w losowe miejsce na ekranie wykonywania aplikacji. Wystarczy, że współrzędna **x** będzie się losowała z liczb od -240 do 240, a **y** od -180 do 180. Żeby program wyglądał zawsze tak samo po uruchomieniu, dodajmy jeszcze skrypt, który sprawi, że po kliknięciu na zieloną flagę kaczka ustawi się w punkcie (0, 0).

7 Sprawdźmy działanie programu. Celowanie możemy ułatwiać i utrudniać, na przykład zmieniając wielkość duszków. Możemy też pójść dalej – rozmiar duszka będzie losowany, a trafienia w mniejszą kaczkę będą wyżej punktowane niż trafienia w dużą. Ciekawym pomysłem jest też dodanie efektów dźwiękowych – dla kaczki i dla broni oraz sygnału trafienia lub dla pudła.

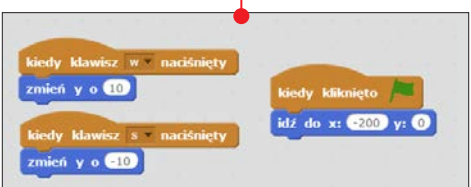
Gra 3: Pong



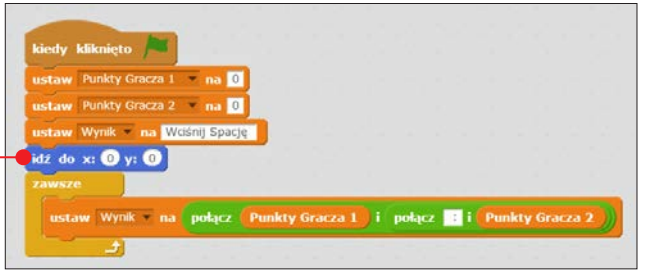
Pong to dwuwymiarowy symulator tenisa sportowego. Gracz, poruszając prostokątem symulującym paletkę, stara się zdobyć punkt poprzez posłanie piłki obok prostokąta drugiego gracza. Była to jedna z pierwszych gier komputerowych i konsolowych na świecie. Teraz postaramy się ją odtworzyć w Scratchu. Nie będzie to wcale trudne.

1 Przygotujmy najpierw wszystkie elementy graficzne naszej gry. Ustawmy kolor sceny na ciemnozielony. Prawą i lewą krawędź tła pokolorujmy jednak inaczej – tych miejsc program będzie używał do punktowania graczy. Oprócz tego musimy stworzyć jeszcze trzy duszki – dwa prostokąty, czyli paletki, i jedno koło, które będzie piłką. Dodatkowo stwórzmy trzy zmienne, które będą zajmowały się obsługą wyników – **Punkty Gracza 1**, **Punkty Gracza 2** i **Wynik**.

2 Pora przejść do zaprogramowania ruchu duszków – paletek. Musimy zagwarantować, że po kliknięciu na zieloną flagę będą się ustawiały zawsze w tym samym miejscu po obu stronach planszy. Trzeba je zaprogramować tak, żeby przy naciskaniu odpowiednich klawiszy na klawiaturze mogły poruszać duszkiem – paletką w górę i w dół. Dla pierwszej paletki może być to ↑ i ↓, a dla drugiej na przykład litery W i S.



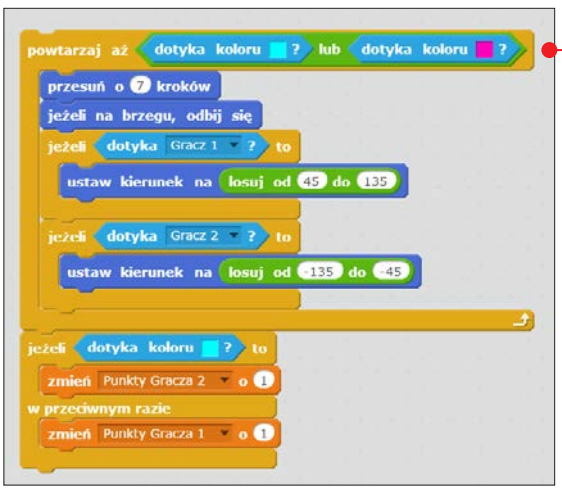
Obie paletki kodujemy tak samo, inne będą tylko klawisze (W i S oraz ↑ i ↓)



3 Teraz przejdźmy do zakodowania zachowania piłki. To ona będzie odpowiedzialna za przypisywanie punktów i za ruch. Najpierw zajmiemy się czynnościami, jakie program musi wykonać na początku. Wraz ze startem programu piłka będzie się ustawiała w punkcie (0, 0) ●. Trzeba wyzerować też punktację obu paletek, a zmienną **Wynik** ustawić tak, by prezentowała na ekranie wykonywania aplikacji punkty jednego i drugiego zawodnika.

4 Gra będzie się zaczynała, gdy program przechwyci wciśnięcie klawisza spacja. Po tym zdarzeniu powinna się rozpocząć pętla trwająca, aż któryś z graczy zdobędzie pięć punktów ●. Następnie piłka powinna ustawić się na środku ekranu wykonywania aplikacji oraz odczekać dwie sekundy. Przerwę wstawiamy, by użytkownicy mieli chwilę na przygotowanie się do gry między dwoma zagrywkami. Inaczej zabawa byłaby bardzo utrudniona. Kierunek duszka ustawmy na losową wartość tak, by żaden z graczy nie wiedział, w którą stronę poleci piłka.

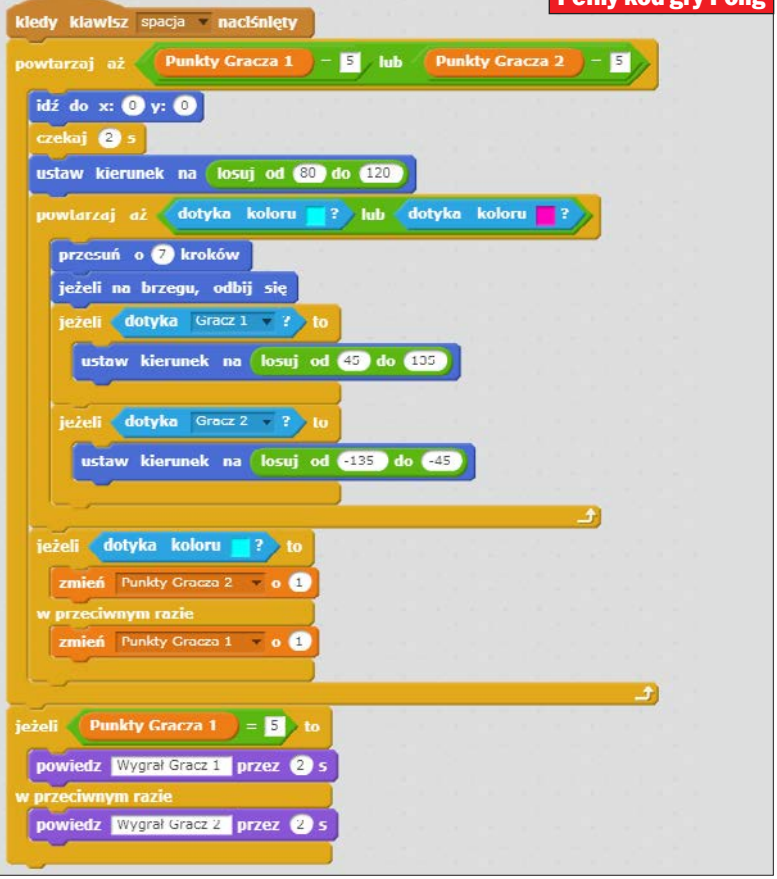
5 Wewnątrz tej pętli musimy teraz wstawić drugą, która po każdym przesunięciu będzie sprawdzała, czy piłka nie dotknęła jednego z kolorów przy prawej lub lewej krawędzi ●. Musimy też zagwarantować, że piłka będzie się odbijała od krawędzi górnej i dolnej oraz od samych paletek. Ta część wymaga trochę poeksperymentowania, ale im więcej pracy



w nią włożymy, tym lepszy i bardziej realistyczny efekt uzyskamy. Na końcu wstawmy jeszcze instrukcję warunkową, która doda punkt jednemu z graczy.

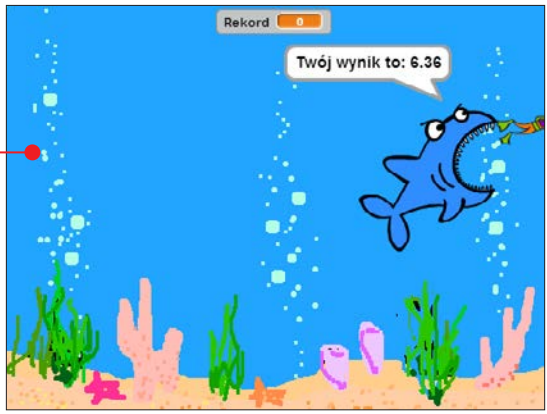
6 Gra jest już na ukończeniu. Po zakończeniu głównej pętli powinniśmy jeszcze dodać instrukcję warunkową, która będzie mówiła graczom, któremu z nich udało się wygrać. Tym samym skończymy kodowanie naszej gry. Jeśli chcemy, możemy teraz ulepszać działanie programu poprzez przyspieszanie ruchu piłki czy nawet wprowadzenie możliwości regulowania prędkości jej ruchu przez użytkowników. Dużym wyzwaniem jest natomiast wprowadzenie możliwości zagrania z komputerowym przeciwnikiem.

Pełny kod gry Pong



Gra 4: Atak rekina

Teraz stworzymy grę zręcznościową ze stale zwiększającym się poziomem trudności. Będą w niej występowały dwie postaci – nurek i rekin. Znajdziemy je w bazie kostiumów. Także podwodną scenerię weźmiemy z biblioteki Scratcha. Zadaniem rekina będzie jak najszybsze dotarcie do nurka, a zadaniem nurka – jak najszybsza ucieczka. Rekin będzie jednak stale zwiększał swoją prędkość – to dzięki temu gra będzie mogła się skończyć. Najlepszy ze zdobytych wyników z danej sesji zostanie zapisany jako rekord.



1 Gdy dodamy już wszystkie elementy graficzne, możemy zacząć kodować. Zacznijmy od nurka. Jego zadaniem będzie „trzymanie się" myszy podczas ucieczki przed rekinem. Dołożymy do tego również opcje uruchamiające grę – rekin będzie startował, gdy klikniemy na nurka. Ostatnim zadaniem postaci będzie też zerowanie stopera, pozwalające później na zmierzenie czasu ucieczki. Gdy klikniemy na zieloną flagę, nurek powinien też ustawiać się w wybranym miejscu oraz informować, że rozpoczęcie gry wymaga kliknięcia na nurka.

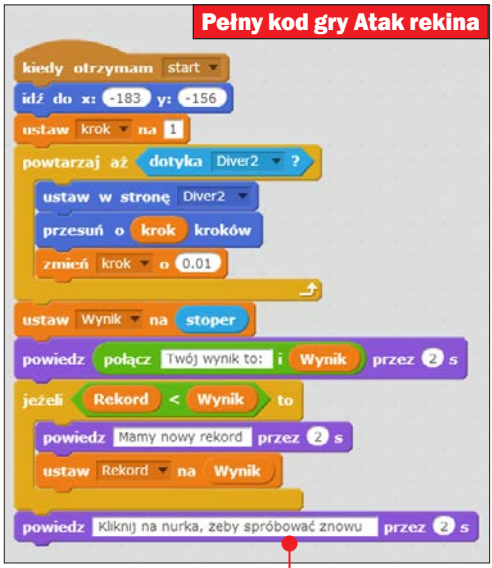
2 Przejdźmy teraz do rekina. Do oskryptowania go będziemy potrzebowali trzech zmiennych, nazwijmy je: **Rekord**, **Wynik** i **Krok**. Zadań pierwszych dwóch nie trzeba tłumaczyć. Trzecia z nich będzie przechowywała liczbę kroków, o jaką będzie się poruszał nasz duszek.

Pozwoli nam to na jej zwiększanie, a tym samym utrudnianie zadania graczowi. Gdy zmienne będą dodane, wstawimy jeszcze instrukcje ustawiające rekina w lewym dolnym rogu ekranu, zwróconego ku prawej stronie, oraz bloczek zerujący zmienną **Rekord**.

3 Wcześniej wstawiliśmy do kodu nurka instrukcję **Nadaj start**. Wykorzystamy to teraz w uruchomieniu rekina. Dzięki temu po odnotowaniu kliknięcia na nurka rekin od razu przejdzie do pościgu. Najpierw jednak wstawmy bloczki ustawiające rekina w lewym dolnym rogu ekranu i zmienną **Krok** na 1. Od teraz po każdej skończonej sesji rekin wróci na

miejsce, a jego prędkość zostanie zmieniona na ustawienia początkowe, dając w ten sposób każdemu z graczy równe szanse. Dalej powinniśmy wstawić pętlę, która będzie działała, dopóki rekin nie dotknie nurka. Wewnątrz pętli każdorazowo nasz duszek powinien ustawić się w stronę pływaka oraz przesunąć się o taką liczbę kroków, jaka zapisana jest w zmiennej **Krok**. By prędkość po każdym przejściu pętli się zwiększała, dodajmy bloczek zwiększający wartość tej zmiennej o 0,01. To bardzo niewiele, ale przejść pętli będzie bardzo

Pełny kod gry Atak rekina

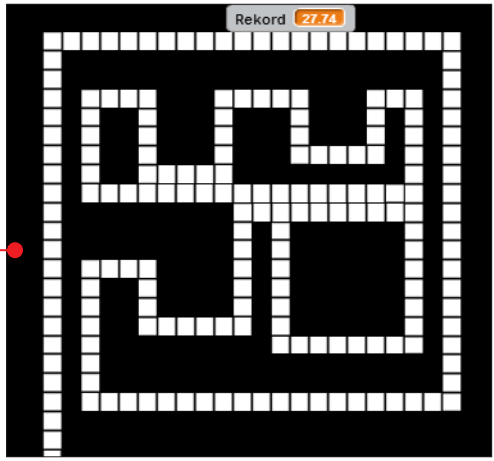


dużo. Już po minucie rekin zacznie naprawdę bardzo szybko pływać.

4 Powyższa pętla zatrzyma się, gdy rekin zetknie się z nurkiem. Gdy to się stanie, program powinien podać wynik oraz sprawdzić, czy nie jest to przypadkiem nowy rekord. Jeśli tak jest, należy pogratulować graczowi oraz poinformować, że jeśli chciałby spróbować jeszcze raz, musi ponownie kliknąć na nurka.

5 Ulepszenia, jakie możemy dodać, to efekty dźwiękowe, przeszkody i tablica wyników. Byłyby na niej wyświetlane imiona pięciu najlepszych zawodników oraz ich czasy. Zawodnicy musieliby w takim wypadku podawać swoje imiona. Lista mogłaby pokazywać się w przerwie między sesjami gry. Można z powodzeniem użyć do tego list. To bardzo dobre ćwiczenie, które zajmie tylko chwilę pracy.

Gra 5: Tron

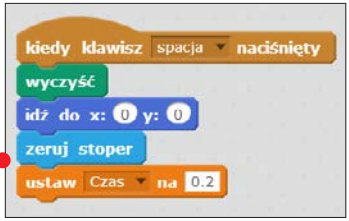


Ta gra będzie wzorowana na klasycznej zręcznościowej grze wideo Tron. Gracz będzie prowadził po planszy pikselowy pojazd zostawiający za sobą ślad – i ma to robić, jak najdłużej zdoła. Trudność polega na tym, że w momencie najechania na własny ślad lub na krawędź ekranu wykonywania aplikacji jazda się kończy. Ważny jest oczywiście czas oraz umiejętność szybkiego wciskania klawiszy, gdyż duszek będzie ciągle przyspieszał. Standardowo będzie też wprowadzone pole przechowujące aktualny rekord.

1 Zaczniemy od stworzenia odpowiedniego duszka. Powinien to być kwadrat niewielkich rozmiarów – im większy, tym mniejsza będzie szansa na zdobycie dobrego rekordu. Za mały duszek spowoduje z kolei, że gra może się dłużyć. Gdy już to zrobimy, dodajmy obsługę sterowania kierunkami, w jakie zwrócony jest duszek. By program był intuicyjny, powinno się to dziać z użyciem strzałek na klawiaturze. Dodajmy również trzy zmienne potrzebne do wykonywania programu: **Czas**, **Wynik** i **Rekord**.

2 Na razie nasza gra jeszcze nic ciekawego nie robi. Zmieni się to teraz, gdy sprawimy, by po naciśnięciu klawisza spacja duszek został wprawiony w ruch. Najpierw musimy jednak zająć się wstępem do działania programu. Ma on być używany wielokrotnie w czasie jednej sesji, a więc po naciśnięciu spacja:

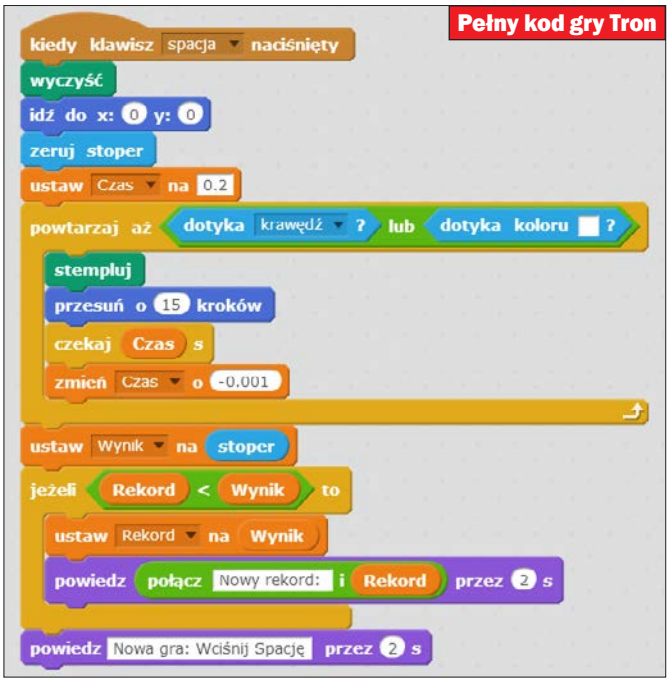
- plansza powinna być wyczyszczona z pozostałości po wcześniejszych grach,
- duszek powinien ustawić się w punkcie (0, 0),
- stoper ma zostać wyzerowany – od tego momentu zacznie się mierzenie czasu gracza,
- zmienna rekord powinna być ustawiona na wartość 0.2 (Scratch używa kropki jako separatora dziesiętnego, a nie przecinka).



3 Pora przejść teraz do zakodowania sedna naszego programu odpowiedzialnego za główną część rozrywki. Będzie to pętla, która będzie działać aż do zetknięcia z kolorem duszka lub krawędzią sceny. Po każdym przejściu pętli zmienna **Czas** będzie też zmniejszana o 0,001 s. Dzięki temu sprawimy, że szybkość przemieszczania się duszka będzie stale rosła. Nasz „pojazd" będzie zostawiał ślad dzięki wstawieniu bloczka **stempluj**. Chwilę uwagi musimy poświęcić na liczbę kroków, o jaką będzie się przemieszczał duszek. Jeśli ustawimy jej wartość zbyt niską, to program może nie zadziałać. Dlaczego? Za mały odstęp między duszkiem a jego śladem sprawi, że program odnotuje zetknięcie z kolorem śladu i tym samym zakończy pętlę. Ten etap wymaga trochę poeksperymentowania. Warto jednak zaczynać od wartości większych i systematycznie je zmniejszać.

4 Teraz pozostało nam już tylko dodać bloczki powiadamiające gracza o jego wyniku oraz sprawdzające, czy nie uzyskał on rekordowego czasu. Dodajmy też bloczek informujący, że ponowne wciśnięcie spacja rozpocznie nową grę. Ważne też jest, by sprawdzić, czy nasza gra nie zawiera błędów. Jeśli tak będzie, to możemy spróbować zmniejszyć rozmiar kwadratu lub zwiększyć liczbę kroków duszka. A potem zostaje już tylko dobra zabawa.

Pełny kod gry Tron



ROZDZIAŁ 6

Zadania z programowania

Dzięki poprzednim rozdziałom poznaliśmy Scratcha już całkiem dobrze. Sprawdźmy teraz naszą wiedzę, wykonując samodzielnie zadania

W tym rozdziale znajdują się zadania, które pozwolą nam sprawdzić nasze umiejętności programistyczne. Zostały one pogrupowane według czterech poziomów trudności – możemy je wybierać w zależności od naszego poziomu zaawansowania. Tematyka ćwiczeń to nie tylko zagadnienia informatyczne. Zamieszczone tu projekty zostały tak dobrane, że powinny spodobać się początkującym informatykom o różnych zainteresowaniach i pomóc im rozwijać kreatywne myślenie. Zadania wraz z przykładowymi rozwiązaniami, które znajdziemy na kolejnych stronach, to nie wszystko – więcej pobierzemy z KŚ+. W KŚ+ znajdziemy też przykładowe skrypty rozwiązań.

DROGOWSKAZ

Poziom 1

Zadanie 1: Przedstaw się

Podaj parę informacji o sobie (imię, wiek, hobby, ulubiona potrawa), korzystając ze Scratcha. Dowolnie zmień swojego duszka i tło sceny. Do stworzenia programu użyj fioletowych bloczków **powiedz... przez... s**.

Przykładowe rozwiązanie:

1 Zaczniemy od ułożenia odpowiedniego skryptu złożonego z fioletowych bloczków **powiedz... przez... s**. W każdym z nich piszemy jedno zdanie o sobie, które spełnia założenia polecenia. Może wyglądać tak:

2 Kolejnym krokiem jest wybranie odpowiedniego tła dla sceny. Takim, które nie będzie odwracało uwagi od działania programu, jest na przykład **doily**.

3 Z kostiumem duszka postąpimy nietypowo. Spróbujemy wgrać własne zdjęcie z komputera. W tym celu na zakładce **Kostiumy** klikamy na **Wgraj kostium z pliku**. Wybieramy plik z fotografią i klikamy na **Otwórz**.

4 Twarz zostanie załadowana wraz z tłem. Teraz użyjemy gumki w edytorze grafiki, by je wymazać. Oto efekt.

Zadanie 2: Podróże małe i duże

Gdzie byłeś? Gdzie chciałbyś pojechać? Jakie projekty możesz stworzyć w Scratchu o miejscach wartych odwiedzenia? Pokaż to za pomocą odpowiedniej aplikacji.

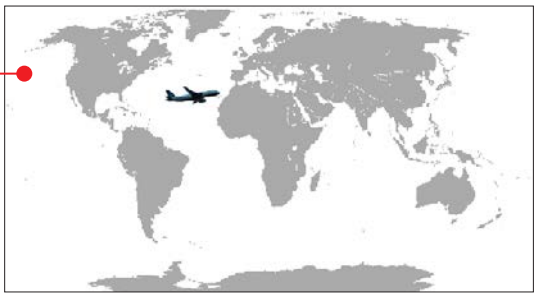
Przykładowe rozwiązanie:

1 Naszym celem będzie stworzenie krótkiej wycieczki po planecie. Podczas tej wyprawy będziemy zachwalać miejsca warte odwiedzenia. Zaczniemy

od dodania odpowiedniego tła. W taki sam sposób, jak w poprzednim zadaniu dodaliśmy swoje zdjęcie, teraz wstawimy własny obrazek tła. Najlepiej – mapę naszej planety.

2 Dodajmy teraz duszka pasującego do motywu podróży. W bazie Scratcha znajdziemy na przykład samolot. Zmniejszymy go trochę, by nie przesłaniał większej części mapy. Po tych dwóch operacjach ekran wykonywania aplikacji powinien wyglądać mniej więcej w ten sposób.

3 Musimy teraz ułożyć odpowiedni kod dla naszego samolotu. Oprzemy go na bloczku **leć przez... s do x... y...**



oraz klocku **powiedz... przez... s**. Duszek będzie przelatywał w określone miejsce. Gdy trafi do celu, powie na temat danego kraju kilka ciekawostek. Współrzędne **x** i **y** danej lokalizacji będziemy mogli odczytać za pomocą wskaźnika myszy. Zakodujemy tak cztery miejsca.

Zadanie 3: Tworzymy scenkę

Za pomocą dwóch duszków i klocków **powiedz... przez... s** i **czekaj... s** zaaranżuj dialog między dwoma postaciami. Tło sceny i wygląd swoich duszków dopasuj do tematyki rozmowy.

Przykładowe rozwiązanie:

1 Nasz projekt wymaga dwóch duszków oraz odpowiedniej sceny. Zgodnie z poleceniem powinny być ze sobą powiązane tematycznie. Wykorzystajmy do tego kostiumy **Parrot** i **Parrot2** oraz tło sceny **beach malibu**.

2 Zaczynamy kodować obydwa duszki. Będziemy do tego używali dwóch bloczków: **powiedz... przez... s** oraz **czekaj... s**. Kluczem do udanego skryptu jest to, że w momencie, gdy jedna z papug będzie mówiła, druga powinna czekać. Oto przykładowy scenariusz rozmowy:

Papuga 1

Mówi: Cześć. Jak masz na imię? (2 s)
Czeka (2 s)
Mówi: Witaj, Bazyl. Co tu robisz?
Czeka (2 s)
Mówi: I jak? Złowiłeś coś?
Czeka (2 s)

Papuga 2

Czeka (2 s)
Mówi: Cześć. Jestem Bazyl. (2 s)
Czeka (2 s)
Mówi: Przyszedłem łowić ryby. (2 s)
Czeka (2 s)
Mówi: Nie. Dziś nie biorą. (2 s)

Pełny kod pierwszej papugi

Pełny kod drugiej papugi

Poziom 2

Zadanie 1: Pensja

Stwórz program, który będzie wyliczał, ile złotych zarobi użytkownik po przepracowaniu podanej liczby godzin. Przyjmij stawkę 13 złotych za godzinę. Na przykład, jeżeli użytkownik poda, że przepracował 10 godzin, to program powinien powiedzieć mu, że zarobił 130 złotych.

Przykładowe rozwiązanie:

1 By program poprawnie obliczał, ile pracownik zarobił, najpierw musimy wiedzieć, ile godzin przepracował. Program musimy zacząć od zadania pytania: **Ile godzin pracowałeś?**

2 Stwórzmy następnie zmienną o nazwie **pensja** równą wartości odpowiedzi pomnożonej przez 13.

3 Teraz pozostaje już tylko wstawić informację zwrotną dla użytkownika. Żeby kwota ładnie wyglądała, możemy użyć do jej zapisania przez program bloczka **połącz... i...**.

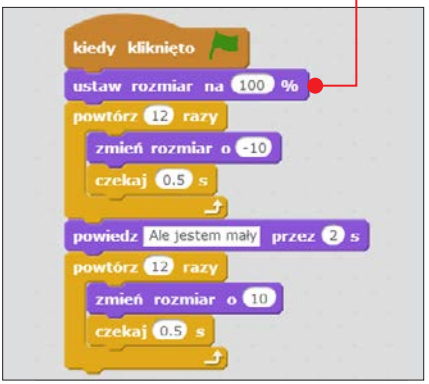
Zadanie 2: Połącz bloczki, by program działał

Połącz wszystkie bloczki tak, żeby kot zmalał, powiedział, że jest mały, a następnie powrócił do swojego poprzedniego rozmiaru.

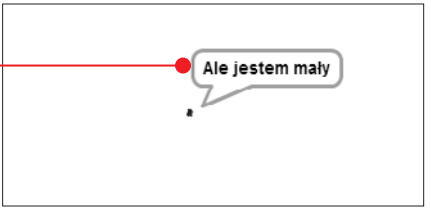
Przykładowe rozwiązanie:

1 Rozwiązywanie najlepiej zacząć od przygotowania wszystkich bloczków, tak by o żadnym z nich nie zapomnieć.

2 Program zaczniemy od ustawienia rozmiaru duszka na jego normalną wielkość. Użyjemy do tego bloczka **ustaw rozmiar na 100 %**.



3 Następnie, za pomocą pętli, zaanimujemy operację zmieniania rozmiaru na mniejszy. Po każdej zmianie rozmiaru o **-10** wstawimy bloczek **czekaj 0.5 s**, żeby lepiej widzieć, jak duszek maleje.



4 Po ukończeniu pętli duszek jest już mały. Musi poinformować o tym użytkownika. Gdy to zrobi, układamy identyczną jak przedtem pętlę, ale ze zmianą rozmiaru o **10**. Sprawimy w ten sposób, że duszek będzie stopniowo zwiększał swój rozmiar.

Zadanie 3: Co jest nie tak z tym programem?

Tadek chciałby, żeby w jego programie kot najpierw podskakiwał w górę i w dół, a potem poruszał się, biegnąc wzdłuż ekranu. Kot skacze jednak do przodu. Co się dzieje? Dokonaj odpowiednich zmian w kodzie, by program działał.

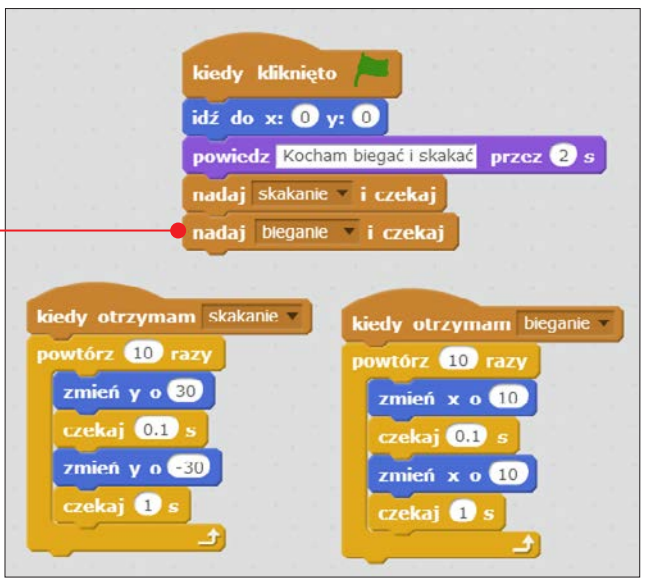
Przykładowe rozwiązanie:

1 Zacznijmy od ułożenia kodu tak, jak ma to miejsce na rysunku powyżej.

2 Włączmy aplikację i sprawdźmy. Przekonamy się, że kot faktycznie wykonuje skoki do przodu. Jeśli natomiast dwukrotnie kliknęlibyśmy na jedną z instrukcji **kiedy otrzymam...**, to zostanie ona wykonana poprawnie. To nie one stanowią więc problem.

3 Okazuje się, że problemem są bloczki **nadaj...**. Sprawiają one, że sygnał na skakanie i bieganie jest wydawany w tym samym momencie, a więc duszek wykonuje obie te rzeczy naraz.

4 Aby kod działał poprawnie, trzeba podmienić te bloczki na **nadaj... i czekaj**. Dzięki temu skrypt rozpoczynany po kliknięciu na zieloną flagę będzie pauzował, aż kot skończy bieganie, i dopiero wtedy pozwoli mu zacząć skakać.



Poziom 3

Zadanie 1: Liczby parzyste i nieparzyste

Stwórz program sprawdzający, czy liczba podana przez użytkownika jest liczbą parzystą czy nieparzystą.

Przykładowe rozwiązanie:

1 Program musi ustalić parzystość lub nieparzystość liczby podanej przez użytkownika. Musimy więc najpierw pobrać od niego tę liczbę. Wstawiamy zatem bloczek **zapytaj... i czekaj** z poleceniem **Podaj liczbę całkowitą**.

2 Kolejnym krokiem jest wstawienie instrukcji warunkowej, która sprawi, że duszek będzie mówił, czy liczba jest parzysta czy nie.

UWAGA!

Dzielenie z resztą jest obsługiwane przez bloczek **...mod...**. Jeśli w skrypcie programu wstawimy instrukcję **5 mod 3**, to program zwróci liczbę 2, bo 5 dzielone przez 3 to jedna całość i 2 reszty.

3 Warunkiem, który będziemy sprawdzali, będzie reszta z dzielenia bloczka **odpowiedź** przez 2. Określimy, czy będzie ona równa 0. Jeśli tak – mamy liczbę parzystą, jeśli nie – nieparzystą. Bloczek obsługujący dzielenie z resztą to **...mod...**.

Zadanie 2: Palindrom

Użytkownik podaje dowolną liczbę trzycyfrową. Zadaniem programu jest sprawdzenie, czy ta liczba jest palindromem (czyli przy czytaniu z lewej do prawej i odwrotnie jest jednakowa).

Przykładowe rozwiązanie:

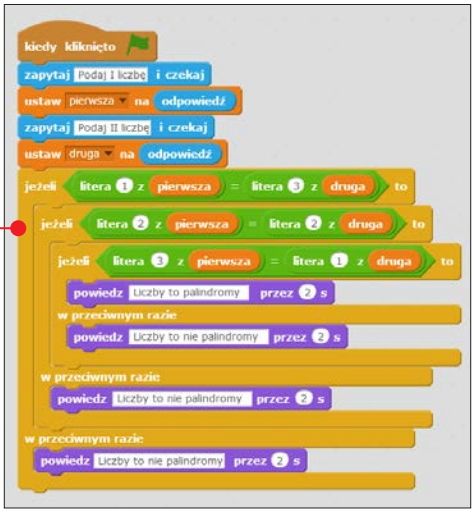
1 Działanie aplikacji musimy zacząć od pobrania od użytkownika dwóch liczb. Zapiszemy je w zmiennych o nazwach **pierwsza** i **druga**.

2 Liczby będą palindromami, jeśli jednocześnie zajdą trzy warunki:

- cyfra setek pierwszej będzie równa cyfrze jedności drugiej,
- cyfry dziesiątek obu liczb będą takie same,
- cyfra jedności pierwszej będzie równa cyfrze setek drugiej.

Musimy zatem stworzyć taką instrukcję warunkową, która będzie sprawdzała, czy jednocześnie zachodzą te wszystkie trzy warunki. Trzeba zapisać każdy z tych warunków, a następnie połączyć je dwoma bloczkami **...i...**.

3 Jeśli taki warunek wydaje nam się zbyt skomplikowany, możemy zamiast niego użyć trzech instrukcji warunkowych, które po kolei będą sprawdzały każdy z wymienionych wcześniej warunków. Muszą one być w sobie zagnieżdżone – jeśli pierwszy warunek zostanie spełniony, to mamy przejść do drugiego, a jeśli nie – od razu powiedzieć, że liczby nie są palindromami.



Zadanie 3: Totolotek

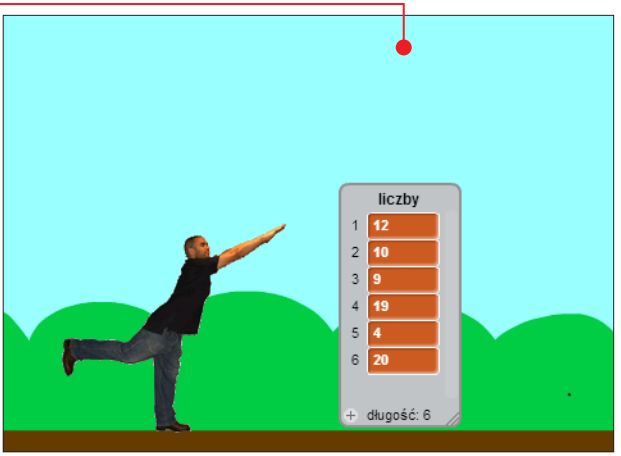
Wykonaj w Scratchu program losujący sześć liczb z przedziału od 1 do 49. Pamiętaj, że nie mogą się powtarzać.

Przykładowe rozwiązanie:

1 Do zapisu losowanych liczb będziemy używali nie zmiennych, ale tym razem listy. Sprawimy w ten sposób, że łatwiej będzie można sprawdzić, czy to, co wylosowaliśmy, nie znajduje się już w naszej bazie. Stwórzmy zatem listę o nazwie liczby i sprawmy, by po kliknięciu na zieloną flagę była ona kasowana. W ten sposób

program po każdym uruchomieniu będzie startował od tego samego momentu.

2 Następnie dodajemy pętlę z sześciokrotną liczbą powtórzeń – po jednym na każdą z losowanych liczb.

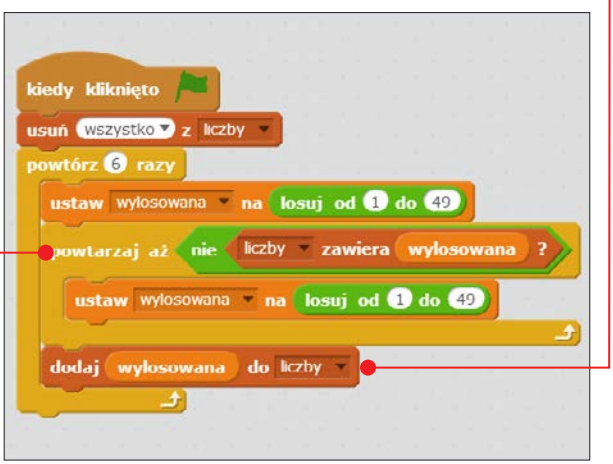


3 Dodajmy zmienną o nazwie **wylosowana** – będziemy jej używali do kontrolowania wylosowanych liczb.

Wstawmy ją w pętli i ustalmy jej wartość na losową liczbę z przedziału od 1 do 49. Skrypt powinien do tej pory wyglądać tak.

4 Wstawmy teraz część skryptu sprawdzającą, czy wylosowana liczba nie pojawiła się już na naszej liście. Będzie to pętla **powtarzaj aż...** z wstawionym warunkiem złożonym z trzech bloczków: **nie...**, **liczby zawiera...** i zmiennej **wylosowana**. W tłumaczeniu warunek będzie sprawdzał, czy na naszej liście o nazwie **liczby** znajduje się wylosowana przez program liczba. Jeśli tak będzie, losowanie musi nastąpić ponownie, aż do momentu, kiedy padnie inny wynik.

5 Gdy będziemy pewni, że wylosowana liczba jest niepowtarzalna, możemy dopisać ją do naszej listy **liczby**.



Poziom 4

Zadanie 1: Liczba pierwsza

Utwórz program sprawdzający, czy podana przez użytkownika liczba naturalna jest liczbą pierwszą. Przyjmujemy, że użytkownik podał liczbę większą od 1.

Przykładowe rozwiązanie:

1 Zaczniemy od pobrania liczby do sprawdzenia. Dzięki założeniu zadania, że użytkownik podaje liczbę większą od 1, nie musimy przejmować się obsługą zdarzeń w przypadku podania zera lub jedynki. Jak wiadomo, te liczby nie są ani liczbami pierwszymi, ani złożonymi.

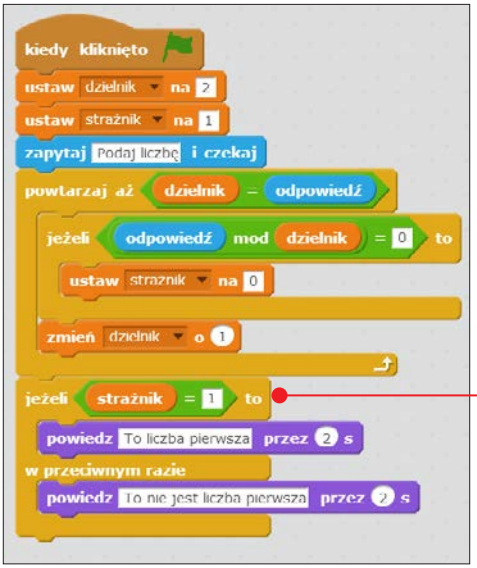
2 Dodajmy zmienną o nazwie **dzielnik** i ustawmy jej wartość – w momencie uruchamiania programu kliknięciem na zieloną flagę – na 2. Za pomocą tej zmiennej będziemy sprawdzali po kolei podzielność liczby podanej przez użytkownika przez kolejne liczby naturalne. (Nie ma sensu sprawdzać, czy 1 jest dzielnikiem odpowiedzi – 1 jest dzielnikiem każdej liczby naturalnej).

UWAGA!

Jeśli liczba ma inny dzielnik niż ona sama i jedynka, to znaczy, że nie jest to liczba pierwsza.

3 Stwórzmy też zmienną o nazwie **strażnik**, za pomocą której ocenimy, czy liczba podana przez użytkownika jest liczbą pierwszą, czy nie. Jeśli w dzieleniu przez kolejne liczby naturalne zmiennej **odpowiedź** przez zmienną **dzielnik** reszta będzie równa 0, to stażnik zmieni swoją wartość na 0. W przeciwnym wypadku będzie ona do końca programu równa 1 – co będzie oznaczało, że mamy do czynienia z liczbą pierwszą. Wartość zmiennej **strażnik** ustalmy na 1 – po kliknięciu na zieloną flagę.

4 Wstawmy teraz pętlę, która będzie działała, aż zmienna **dzielnik** będzie równa zmiennej **odpowiedź**. Musimy w niej zawrzeć instrukcję warunkową sprawdzającą, czy wartość reszty z dzielenia **odpowiedzi** przez **dzielnik** jest równa 0. Jeśli tak będzie, to oznacza, że użytkownik nie podał liczby pierwszej. By sprawdzić podzielność przez wszystkie liczby mniejsze od odpowiedzi, zmienna dzielnik musi się zwiększać o 1 po każdym przejściu pętli.



UWAGA!

Przedstawiony tu algorytm można udoskonalić. Zauważmy, że nie trzeba sprawdzać dzielników aż do momentu równości ze zmienną **odpowiedź**. Po chwili namysłu można stwierdzić, że wystarczy je sprawdzić aż do wartości będącej pierwiastkiem z liczby użytkownika. Zaimplementowanie takiego rozwiązania sprawi, że program będzie działał szybciej, szczególnie dla dużych liczb.

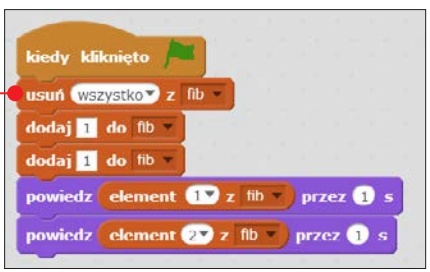
5 Program zakończymy instrukcją warunkową sprawdzającą wartość zmiennej **strażnik**. Jeśli jest ona równa 1, program powinien powiedzieć, że to liczba pierwsza, a jeśli wartość wynosi 0 – że złożona.

Zadanie 2: Ciąg Fibonacciego

Stwórz program wyświetlający kolejne liczby ciągu Fibonacciego.

Przykładowe rozwiązanie:

1 Działanie naszego programu oprzemy na listach. Stwórzmy więc listę o nazwie **fib**. W jej kolejnych rekordach będą przechowywane następne elementy ciągu. Algorytm ciągu wymusza na nas ustalenie pierwszych dwóch elementów na wartość 1. Dodajmy także na początku bloczek kasujący całą listę tak, aby program mógł być uruchamiany wielokrotnie.



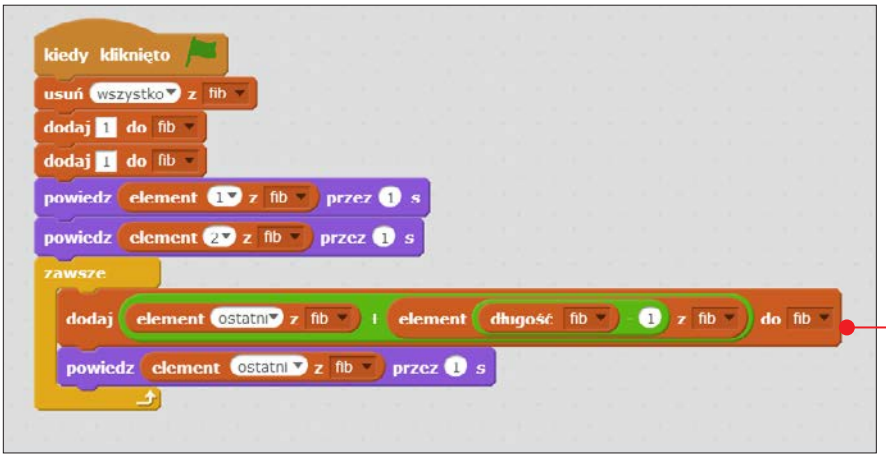
CIĄG FIBONACCIEGO

Ciąg Fibonacciego to ciąg liczb 1, 1, 2, 3, 5, 8, 13, 21..., w którym każda kolejna liczba jest sumą dwóch poprzednich.

2 Wstawiamy pętlę **zawsze**. Będzie miała za zadanie obliczanie kolejnych elementów ciągu. Musimy w niej umieścić instruk-

cję powodującą dodanie nowego rekordu do listy (na ostatnim miejscu), którego wartość będzie sumą dwóch poprzednich ●.

3 Gdy nowy element zostanie dodany do listy, duszek musi go wypowiedzieć ●. Program będzie teoretycznie działał w nieskończoność. Jednak w każdej chwili możemy przerwać jego obliczenia, klikając na czerwony znak **Stop** nad ekranem wykonywania aplikacji.



Zadanie 3: Anagram

Stwórz aplikację, która będzie sprawdzała, czy dwa podane przez użytkownika wyrazy to anagramy. (Ta aplikacja wykorzystuje algorytm wymagany na egzaminie maturalnym z informatyki na poziomie rozszerzonym).

Przykładowe rozwiązanie:

1 Kodowanie aplikacji zaczynamy oczywiście od pobrania wyrazów wybranych przez użytkownika do sprawdzenia. Zapiszmy je do dwóch zmiennych: **wyr1** i **wyr2**.

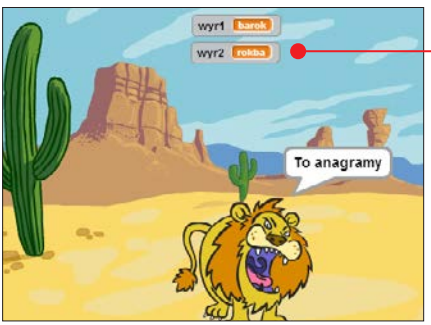
2 Zgodnie z definicją anagramu oba wyrazy muszą mieć dokładnie te same litery. Zatem jeśli nie będą miały tej samej długości, to możemy od razu poinformować użytkownika, że to nie anagramy.

3 Zajmijmy się teraz przypadkiem, gdy obydwa wyrazy mają taką samą długość. Musimy je podzielić na poszczególne litery. Wykorzystamy do tego listy. Każda literka będzie w nich osobnym rekordem. Musimy zatem stworzyć dwie listy – **wyraz1** i **wyraz2** oraz zmienną pomocniczą o nazwie **literka** z początkową wartością ustaloną na 1. Będzie ona przechowywała numer aktualnie przekładanej litery z każdego z wyrazów. Dodajmy też pętlę, która to zadanie wykona. Ma ona mieć tyle powtórzeń, ile będzie liter w wyrazach, a po każdym przejściu zmienna literka powinna zmienić się o 1.

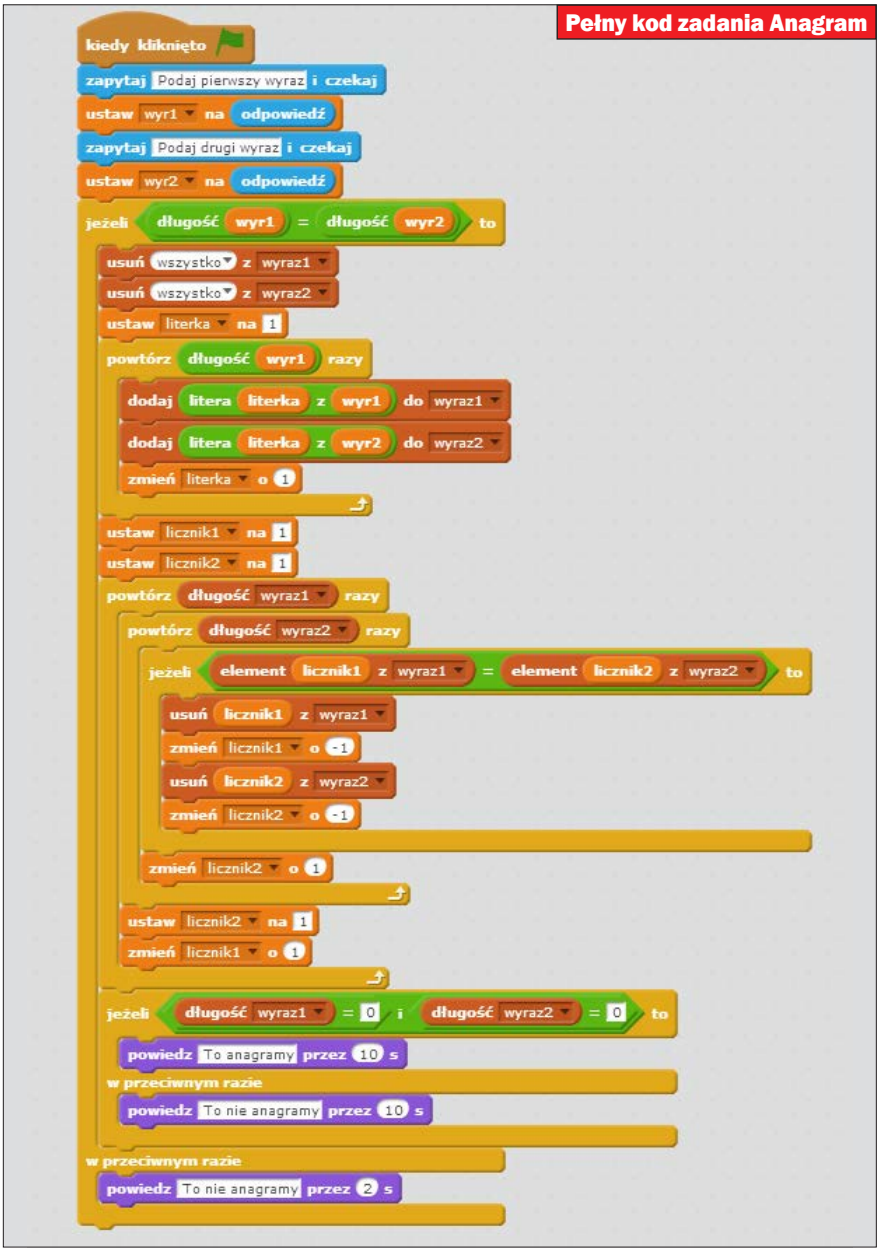
UWAGA!

Skasowanie jednego elementu z listy powoduje przesunięcie następnych rekordów o jedno miejsce wcześniej.

4 Uruchamiając program, zobaczymy, że wyrazy w każdej z list są już prawidłowo podzielone na litery. Teraz pora sprawdzić, czy każda z list zawiera takie same litery i to w takiej samej ilości. Stworzymy do tego dwie pętle, z czego jedna będzie zawarta w drugiej. Sprawdzanie odbędzie się litera po literze. Jeśli algorytm napotka w każdej z list po takiej samej literze, to usunie je z nich. W ten sposób jeśli pod koniec działania skryptu obie listy będą puste, to będzie znaczyło, że użytkownik podał wyrazy, które są anagramami.



Pełny kod zadania Anagram



Dodatkowe zadania

Oto dodatkowe zadania z programowania. Możemy je pobrać i wydrukować. Aby było trudniej, ich rozwiązania nie zostały już podane w formie wskazówek krok po kroku. Za to znajdziemy w KŚ+ przykładowe skrypty. Po skopiowaniu ich na dysk i uruchomieniu w edytorze Scratch zobaczymy, jak należy rozwiązać zadania.

Poziom 1

Zadanie 4: Haiku

Haiku to wiersz składający się z siedemnastu sylab zapisanych w trzech linijkach tekstu (pięć w pierwszej, siedem w drugiej i pięć w trzeciej). Zaanimuj haiku w Scratchu za pomocą trzech bloczków **Powiedz... przez... s**. Pamiętaj, że możesz stworzyć różne postacie i scenerie.

Zadanie 5: Reklama Scratcha

Jak mogłaby wyglądać reklama Scratcha? Jak przekonałbyś kogoś do jego wypróbowania? Stwórz odpowiedni program.

Zadanie 6: Obrazowanie słów

Jakie projekty w Scratchu możesz stworzyć, by pomóc innym zrozumieć takie słowa, jak: *pętla* (czyli powtarzana wielokrotnie czynność), *inny, trudny, czekaj, stój, obcy, piękny, przyjaciel, deszcz, otwarty, miłość, sztuka, syrena, instrukcje, łatwe, moc, czas, bohater.*

Zadanie 7: Kartka świąteczna

Zaprojektuj interaktywną kartkę świąteczną. Stwórz w tym celu własnego duszka i tło. Nie zapomnij o dodaniu dźwięków.

Zadanie 8: Dziękuję

Czy ktoś ostatnio zrobił dla ciebie coś dobrego? Czy jesteś komuś za coś wdzięczny? Jak mógłbyś powiedzieć „dziękuję” za pomocą projektu stworzonego w Scratchu?

Zadanie 9: Pohałasujmy!

Stwórz w Scratchu projekt, zawierający interesujące, nietypowe i inspirujące dźwięki lub muzykę. Możesz w tym celu wykorzystać bloczki przedstawione na rysunku obok.

Zadanie 10: Prostokąty

Nie ma kół i okręgów, nie ma linii, nie ma też importowanych obrazków. Stwórz projekt, wykorzystując jedynie własnoręcznie narysowane prostokąty (tło sceny powinno być również wykonane z prostokątów).

Zadanie 11: Za dużo, za mało

Stwórz program, w którym użytkownik będzie zgadywał liczbę losowo wybraną przez komputer. Będziesz potrzebował duszka, który powie użytkownikowi, że pomyślał o pewnej liczbie od 1 do 100, a następnie poprosi go o odgadnięcie tej liczby. Duszek powinien mówić „Za dużo" i „Za mało" w momencie, gdy użytkownik będzie źle zgadywał. Gdy użytkownik odgadnie wylosowaną liczbę, duszek powinien zmienić wygląd. Skorzystaj z bloczków widocznych na rysunku poniżej.

Zadanie 12: 12 wyzwań

Poniżej znajduje się lista dwunastu wyzwań, każda grupa losuje dla siebie jedno z nich, a następnie stara się je jak najlepiej wykonać.

1. Gdy naciśniesz klawisz B – duszek się nieco zwiększy, a gdy naciśniesz S – zmniejszy.
2. Gdy duszek usłyszy głośny dźwięk, powinien zmieniać swój kolor.
3. Gdy duszek jest w górnej ćwiartce sceny, ma mówić: „Podoba mi się tu na górze”.
4. Gdy duszek dotyka czegoś niebieskiego, zaczyna grać wysoki dźwięk, a gdy dotyka czegoś czerwonego – niski dźwięk.
5. Gdy dwa duszki się zderzają, jeden z nich ma mówić: „Przepraszam”.
6. Zawsze gdy kot i pies się do siebie zbliżają, kot ucieka od psa.
7. Gdy klikniesz na tło sceny, pojawi się kwiatek w tym miejscu.
8. Gdy klikniesz na duszka, drugi duszek zacznie tańczyć.
9. Duszek będzie spadał, jakby działała na niego grawitacja, ale zatrzyma się, gdy dotknie zielonego podłoża.
10. Duszek podąża za kursorem myszy, ale nigdy się do niego nie zbliża.
11. Duszek podąża za czerwoną linią.
12. Gdy wynik osiągnie 10, tło sceny się zmienia.

Zadanie 13: Kwadrat i kółko

Jaki program jesteś w stanie stworzyć, mając do dyspozycji dwa duszki będące pomarańczowym kwadratem i fioletowym kółkiem. Możesz korzystać tylko z poniższych klocków.

Poziom 2

Zadanie 4: Podróżnik

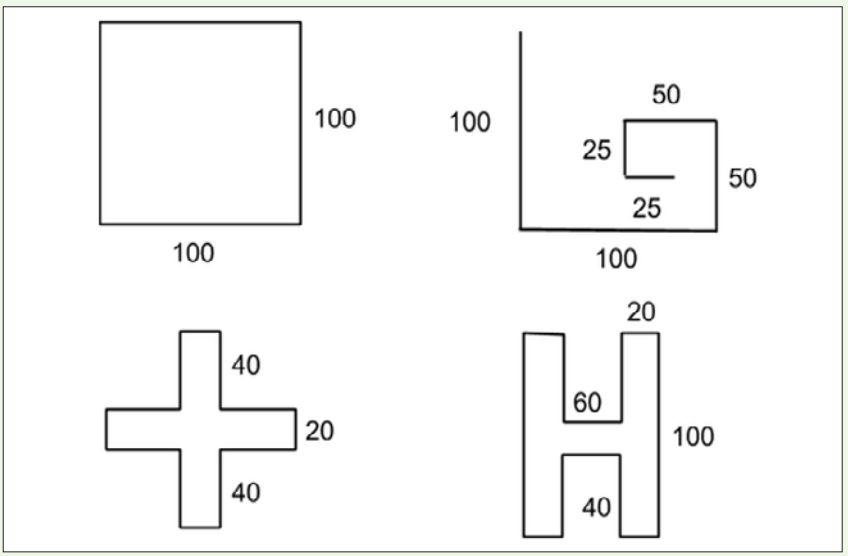
Wykonaj w Scratchu animację opisującą pięć różnych miejsc na świecie. Ustaw tło na mapę świata i spraw, by duszek, przemieszczając się między wybranymi przez ciebie miejscami, powiedział dwa zdania o każdym z nich.

Zadanie 5: Wynalazki

Stwórz program, który będzie prezentował trzy – twoim zdaniem – najważniejsze wynalazki w historii. Zaprezentuj je odpowiednio, używając możliwości, jakie daje Scratch. Użyj dźwięku, animacji duszka i innych obiektów, zadbaj o odpowiedni wygląd sceny.

Zadanie 6: Rysujemy kształty

Narysuj poniższe kształty. Liczby podane obok odcinków to kroki, jakie duszek musi wykonać. Do poruszania duszkiem używaj tylko klocków **Przesuń o... kroków** oraz **Ustaw kierunek na...**.



Zadanie 7: Piszemy swoje imię

Napisz w Scratchu swoje imię. Do poruszania się używaj klocka **Idź do x:... y:...** . Możesz najpierw napisać swoje imię na brudno na arkuszu, po czym odczytać z niego współrzędne, według jakich ma się poruszać duszek.

30,40	-60,40	-40,40	-20,40	0,40	20,40	40,40	60,40	80,4
30,20	-60,20	-40,20	-20,20	0,20	20,20	40,20	60,20	80,2
80,0	-60,0	-40,0	-20,0	0,0	20,0	40,0	60,0	80,(
30,-20	-60,-20	-40,-20	-20,-20	0,-20	20,-20	40,-20	60,-20	80,-2
30,-40	-60,-40	-40,-40	-20,-40	0,-40	20,-40	40,-40	60,-40	80,-4
30,-60	-60,-60	-40,-60	-20,-60	(				

Na ilustracji widać fragment arkusza – całość do pobrania w formacie PDF z płyty lub KŚ+.

Zadanie 8: Sztuka z 12 klocków

Stwórz rysunek, używając tylko tych 12 klocków, które widzisz obok. Nie musisz używać wszystkich z nich. Możesz dowolnie zmieniać ich wartości.

Zadanie 9: Uciekający duszek

Stwórz program, w którym duszek będzie uciekał przed wskaźnikiem myszy, gdy ta będzie się do niego zbliżała na mniej niż 10 kroków.

Zadanie 10: Co jest nie tak z tym programem? (1)

Arek chciałby, żeby jego kot obracał się dookoła, gdy naciśnie spację, ale kot się nie porusza. Co się dzieje? Dokonaj odpowiednich zmian w kodzie – napraw go tak, by program działał.

Zadanie 11: Co jest nie tak z tym programem? (2)

Stefcia chce, żeby kot startował od środka planszy, a potem poruszał się w prawo i rósł. Program działa po pierwszym kliknięciu na zieloną flagę, ale potem już nie. Co się dzieje? Dokonaj odpowiednich zmian w kodzie – napraw go tak, by program działał.

Zadanie 12: Co jest nie tak z tym programem? (3)

Michał chce, żeby jego kot tańczył w rytm muzyki, ale kot zaczyna tańczyć dopiero wtedy, gdy piosenka się skończy. Na czym polega problem? Dokonaj odpowiednich zmian w kodzie – napraw go tak, by program działał.

Zadanie 13: Co jest nie tak z tym programem? (4)

Karolina chce poruszać kotem: po naciśnięciu strzałki w lewo ma on iść w lewo, a strzałki w prawo – w prawo. Chce też, żeby kot mówił, po której stronie planszy się znajduje. Kot porusza się, ale nie podaje poprawnie swojej pozycji. Co się dzieje? Dokonaj odpowiednich zmian w kodzie – napraw go tak, by program działał.

Zadanie 14: 8 bloczków

Jakie programy możesz stworzyć z poniższych ośmiu bloczków?

Zadanie 15: Połącz bloczki, by program działał (2)

Połącz wszystkie bloczki tak, żeby kot spytał, ile jest 6*8, a potem powiedział, czy odpowiedź użytkownika jest dobra.

Zadanie 16: Połącz bloczki, by program działał (3)

Połącz wszystkie bloczki tak, żeby kot cały czas wymawiał współrzędne wskaźnika myszy.

Na powyższym obrazku w jednym z zielonych bloczków znajduje się kropka – to nie jest błąd.

Zadanie 17: Połącz bloczki, by program działał (4)

Połącz wszystkie bloczki tak, żeby kot rysował czerwony kwadrat.

Zadanie 18: Sto lat

Stwórz program, który będzie grał znaną melodię „Sto lat". Do jego wykonania użyj tylko bloczków przedstawionych poniżej.

Poziom 3

Zadanie 4: Średnia arytmetyczna

Stwórz program obliczający średnią arytmetyczną z dwóch liczb podawanych przez użytkownika.

Zadanie 5: Kalkulator

Stwórz prosty kalkulator, który będzie obliczał sumę, różnicę, iloczyn i iloraz dwóch liczb podanych przez użytkownika.

Zadanie 6: Wartość bezwzględna

Napisz program obliczający wartość bezwzględną z liczby podanej przez użytkownika.

Zadanie 7: Prostokątność trójkąta

Stwórz program sprawdzający, czy trójkąt o bokach podanych przez użytkownika jest trójkątem prostokątnym.

Zadanie 8: Litery

Napisz program, który poinformuje, czy podany przez użytkownika znak to samogłoska, spółgłoska czy coś innego.

Zadanie 9: Silnia

Stwórz program obliczający silnię liczby podanej przez użytkownika.

Zadanie 10: Dodatnia czy ujemna

Napisz program, który będzie sprawdzał, czy podana przez użytkownika liczba jest dodatnia, ujemna czy równa 0.

Zadanie 11: Kolejne potęgi

Stwórz program, który po podaniu przez użytkownika liczby naturalnej będzie wyświetlał jej kolejne potęgi, dopóki użytkownik nie naciśnie klawisza K.

Zadanie 12: Dzielniki

Stwórz program wyznaczający wszystkie dzielniki podanej przez użytkownika liczby naturalnej.

Przykładowy efekt działania programu wyznaczającego dzielniki liczb naturalnych

Zadanie 13: Liczba cyfr

Stwórz program, który dla podanego przez użytkownika numeru określi liczbę jej cyfr.

Zadanie 14: Wzór Herona

Utwórz program wyliczający pole trójkąta na podstawie długości jego trzech boków **a**, **b** i **c**, korzystając ze wzoru Herona:

$$p = \frac{1}{2}(a + b + c)$$

gdzie **p** oznacza połowę obwodu tego trójkąta. Gdy tego dokonasz, spraw, by program przed obliczeniem pola figury sprawdzał, czy z trzech podanych przez użytkownika boków da się zbudować trójkat i, jeśli tak nie będzie, prosił użytkownika o ponowne podanie długości boków.

Zadanie 15: Wirtualny policjant

Stwórz program, który będzie wirtualnym policjantem, rozdającym mandaty (zadbaj również o odpowiednią oprawę graficzną). Ma działać w ten sposób, że użytkownik, po podaniu prędkości, z jaką przekroczył dozwoloną, dostanie informację, jaki wysoki mandat musi zapłacić i ile punktów karnych dostanie. Jako pomocy użyj poniższej tabeli.

PRZEKROCZENIE PRĘDKOŚCI	WYSOKOŚĆ MANDATU	PUNKTY KARNE
do 10 km/h	50 zł	0
11–20 km/h	100 zł	2
21–30 km/h	200 zł	4
31–40 km/h	300 zł	6
41–50 km/h	400 zł	8
powyżej 50 km/h	500 zł	10

Zadanie 16: Rok przestępny

Napisz program, który po podaniu przez użytkownika roku stwierdzi, czy jest to rok przestępny, czy nie. Przed przystąpieniem do ćwiczenia dokładnie dowiedz się, kiedy rok traktujemy jako przestępny.

Zadanie 17: Tynk bawełniany

Jedno opakowanie tynku bawełnianego wystarcza na pokrycie 5 m^2 powierzchni ściany. Stwórz program, który w zależności od podanej przez użytkownika powierzchni ściany wskaże, ile opakowań tynku bawełnianego potrzeba do jej pokrycia. Pamiętaj, że odpowiedź ma być podana w postaci liczby naturalnej (nikt nie kupi 2,5 opakowania).

Zadanie 18: Włosy

Ludzkie włosy rosną około 0,35 mm dziennie. Stwórz program, który po wpisaniu przez użytkownika żądanej przez niego długości włosów zwróci informację, jak długo użytkownik będzie musiał zapuszczać włosy.

Poziom 4

Zadanie 4: Dec na Bin

Stwórz program zamieniający dowolną liczbę naturalną na jej odpowiednik zapisany w systemie binarnym.

Zadanie 5: Bin na Dec

Stwórz program zamieniający dowolną liczbę zapisaną w systemie binarnym na jej odpowiednik zapisany w systemie dziesiętnym.

Zadanie 6: Pierwiastki równania kwadratowego

Napisz program zwracający pierwiastki równania kwadratowego podanego w postaci ogólnej.

Zadanie 7: Liczby pierwsze

Utwórz program wypisujący **n** początkowych liczb pierwszych, gdzie **n** jest liczbą podaną przez użytkownika.

Zadanie 8: Największy wspólny dzielnik

Stwórz program obliczający największy wspólny dzielnik dwóch liczb podanych przez użytkownika.

Efekt działania programu wyznaczającego największy wspólny dzielnik dwóch liczb całkowitych

Zadanie 9: Odległość między punktami

Napisz skrypt, który będzie obliczał odległość między dwoma punktami w kartezjańskim (czyli prostokątnym) układzie współrzędnych. Punkty mają być zadane przez użytkownika.

Zadanie 10: Trójki pitagorejskie

Stwórz program, który z liczb z przedziału 1–100 wybierze trójki liczb pitagorejskich, czyli takich, które spełniają warunki z twierdzenia Pitagorasa (na przykład 3, 4, 5).

Zadanie 11: Sortowanie

Zbuduj aplikację, która dziesięć liczb podanych przez użytkownika posortuje malejąco.

Zadanie 12: Sortowanie

Zbuduj aplikację, która dziesięć liczb podanych przez użytkownika posortuje rosnąco.

Zadanie 13: Zliczanie liter

Napisz program, który zliczy wystąpienia każdej z liter w zdaniu i poda tylko te, które wystąpiły co najmniej raz.

Zadanie 14: Palindrom

Napisz program, który określi, czy wczytany ciąg jest palindromem tekstowym.

Zadanie 15: Najkrótsza odległość

Stwórz aplikację, która z podanych przez użytkownika czterech punktów wyznaczy najkrótszą łączącą je łamaną i pokaże to na ekranie wykonywania aplikacji.

Zadanie 16: Punkty kratowe

Napisz program, który będzie wyznaczał, ile punktów kratowych (czyli takich, których współrzędne w układzie kartezjańskim są liczbami całkowitymi) leży wewnątrz okręgu o środku w punkcie (0, 0) i dowolnym promieniu. Całą operację zwizualizuj na ekranie wykonywania aplikacji.

Zadanie 17: Anagram

Stwórz aplikację, która będzie sprawdzała, czy dwa podane przez użytkownika wyrazy to anagramy.

Zadanie 18: Szyfr Cezara

Zbuduj aplikację, która będzie używała szyfru Cezara (czyli szyfru przestawieniowego) do kodowania i dekodowania wiadomości.

Zadanie 19: Grawitacja

Stwórz program symulujący spadek swobodny. Duszek, po wciśnięciu przez użytkownika klawisza spacja, wylosuje sobie położenie na ekranie wykonywania aplikacji i w realistyczny sposób spadnie na podłoże.

ROZDZIAŁ 7

Nietypowe zastosowania Scratcha

Ze Scratcha można korzystać nie tylko na komputerach. Zobaczmy, co da się zrobić ze Scratchem i Arduino czy Raspberry Pi

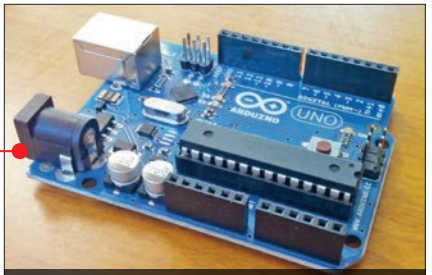
Scratch to nie tylko bloczki i programowanie duszków na komputerze. Gdy już potrafimy tworzyć różne programy, możemy wykorzystywać nasze umiejętności do tworzenia aplikacji dla systemu Android albo spróbować swoich sił z Arduino czy Raspberry Pi.

Wszelkie niezbędne informacje, jak zacząć pracę z każdym z tych systemów, znajdziemy w tym rozdziale.

DROGOWSKAZ

Ujarzmiamy elektronikę – Scratch i Arduino

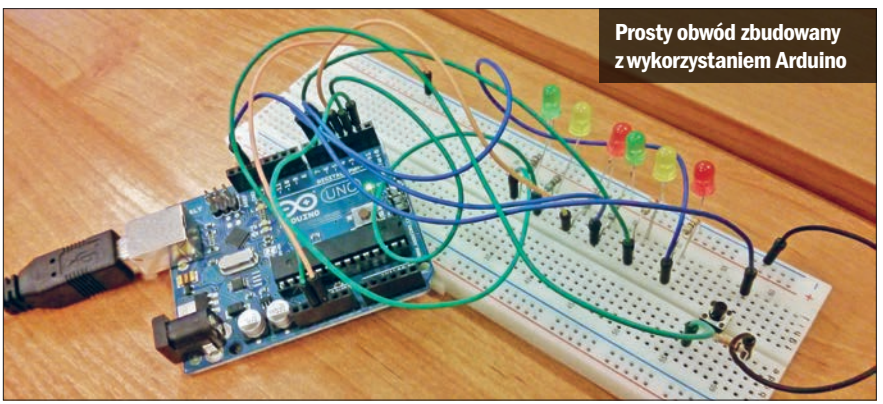
Arduino to popularna platforma programistyczna, która pozwala każdemu, nawet osobom niemającym doświadczenia z programowaniem i elektroniką, tworzyć profesjonalne, inteligentne urządzenia. Powstała w 2005 roku i od tego czasu zaskarbiła sobie przychylność użytkowników. Dzięki Arduino możemy na przykład stworzyć program, który będzie nas informował, kiedy nasze domowe kwiaty potrzebują podlewania. Możemy też ułożyć aplikację wysyłającą SMS na nasz numer telefonu, gdy tylko ktoś otworzy drzwi domu. Nie sprawi nam to większego problemu – jeśli tylko samo programowanie dobrze nam idzie.



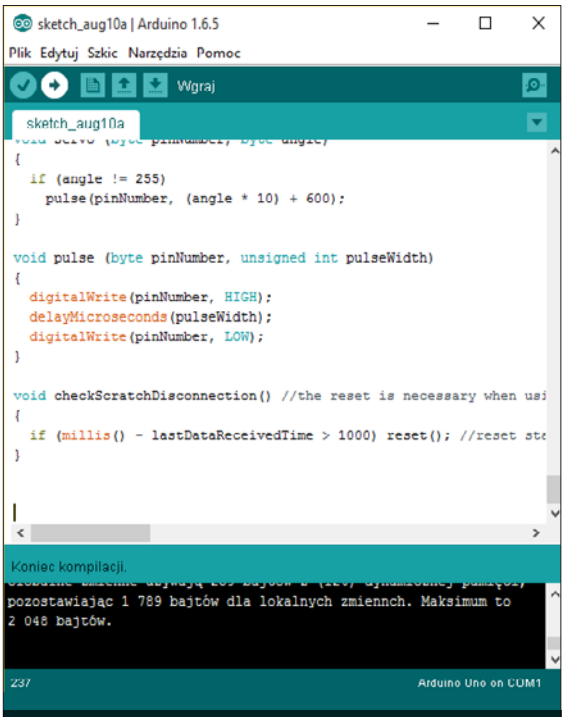
Arduino Uno – najpopularniejsza obecnie wersja kontrolera Arduino

Alarm świetlny

Aby zapoznać się z Arduino, zaprogramujemy alarm świetlny reagujący na kliknięcie w Scratchu. Chcąc stworzyć proste urządzenie oparte na Arduino, musimy wyposażyć się w płytkę Arduino ●. Z powodzeniem możemy znaleźć wiele ofert w internecie. Najlepiej na samym początku również dokupić płytkę stykową, oporniki, diody i kable. To niezbędne minimum, by zacząć pracę. Za wszystkie wspomniane urządzenia (a nawet i więcej) zapłacimy 100–150 złotych. To niewielki wydatek jak na możliwości, jakie dają.


Prosty obwód zbudowany z wykorzystaniem Arduino

Jeśli chodzi o nasz komputer, to powinniśmy wyposażyć go w brata Scratcha – **S4A** (od *Scratch For Arduino*). Nie jest to oficjalna wersja programu, lecz przerobiona wersja podstawowego Scratcha z dodanymi bloczkami, obsługującymi funkcje Arduino.
S4A znajdziemy na płycie dołączonej do książki. Można go też pobrać ze strony o adresie **s4a.cat**. Następnie musimy go zainstalować na naszym komputerze. Potem trzeba zainstalować jeszcze jeden program Arduino. Pozwoli on nam na wgranie odpowiedniego pliku, który połączy ze sobą Scratcha i Arduino – to tak zwany firmware.



Okno programu Arduino z wgranym plikiem firmware

Znajdziemy go na płycie lub pobierzemy: **www.arduino.cc/download_handler.php**. Pozostało jeszcze pobranie pliku, który wgramy na płytkę, i możemy zacząć tworzyć pierwszy program dla Arduino. Plik także znajdziemy na płycie lub pod adresem **vps34736.ovh.net/S4A/S4AFirmware16.ino**. Ten link znajduje się też na stronie **s4a.cat**.
Teraz wykonujemy kolejne kroki:

1 Podłączamy płytkę do komputera kablem USB.

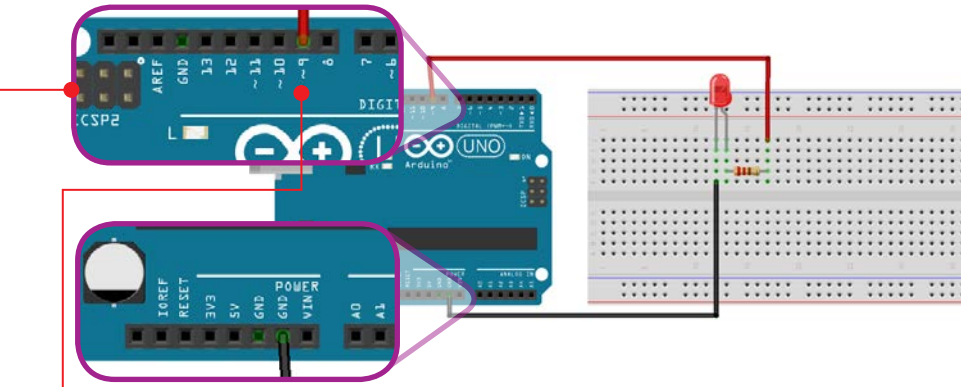
2 Włączamy program Arduino na naszym komputerze. Otwieramy w programie pobrany plik o rozszerzeniu **.ino**.

3 W menu **Narzędzia** wybieramy typ naszej płytki i port, do którego jest podłączona.

4 Ładujemy firmware na płytkę, klikając na strzałkę z napisem **Wgraj**. Czekamy, aż operacja się zakończy, i zamykamy program. Oprogramowanie jest już wgrane. Tę czynność wystarczy wykonać tylko raz.

5 Podłączmy komponenty potrzebne do działania programu. Potrzebujemy jednej diody, rezystora 220 Ω i dwóch kabli. Wszystkie te elementy podłączamy do płytki stykowej według schematu ● (dłuższą nóżkę diody LED podłączamy do rezystora).

6 Jeżeli mamy już wszystko podłączone, przyszła pora na ułoże-



nie odpowiedniego skryptu sprawiającego, że dioda będzie świecić. Otwieramy program **S4A** i tworzymy skrypt, który sprawi, że po kliknięciu na zieloną flagę dioda, którą podłączyliśmy do wyjścia numer 9, zapali się. Skrypt powinien wyglądać jak poniżej.

7 Teraz zobaczmy, jak łatwo rozbudowywać programy dla Arduino, umiejąc już programować w Scratchu. Dokładając tylko trzy bloczki, możemy sprawić, że dioda będzie migać. To proste. Wartość 255 mówi, że dioda ma świecić, a wartość 0 będzie sprawiała, że zgaśnie. Jeśli teraz te bloczki ułożymy w pętli i po każdym przejściu dodamy jedną sekundę przerwy, to nasza dioda zacznie migać.

PRZYDATNE PORADY

- Kabel idący od diody do pola GND, czyli masy (połączenie przeciwporażeniowe), powinien być czarny. Taki jest zwyczaj.
- Dioda ma dwie nóżki: krótszą i dłuższą. Dłuższa powinna przechodzić do masy, inaczej dioda nie będzie świecić.
- Nieważne, gdzie w obwodzie elektronicznym podłączymy opornik. Ważne, żeby był w tym obwodzie. W przeciwnym wypadku dioda może się spalić.
- Schematy Arduino można budować w darmowym programie **Fritzing** (znajdziemy go na płycie dołączonej do książki).
- Wyjściom cyfrowym 5, 6 i 9 na płytce możemy przypisywać wartości od 0 do 255, co pozwala na sterowanie jasnością świecenia diody.
- W wypadku innych wyjść można sprawiać tylko, by działały lub nie, czyli na przykład, aby dioda świeciła lub nie.

Arduino i dioda RGB

Do kolejnego projektu będziemy potrzebowali następujących komponentów:

- Arduino,
- płytki stykowej i przewodów,
- diody RGB – czyli jednej diody, która może świecić na trzy podstawowe kolory: czerwony (R), zielony (G) i niebieski (B),
- trzech rezystorów 220 Ω.

Stworzymy program, który w zależności od tego, jaki klawisz wciśniemy na klawiaturze, będzie zapalał diody RGB w różnych kombinacjach kolorystycznych.

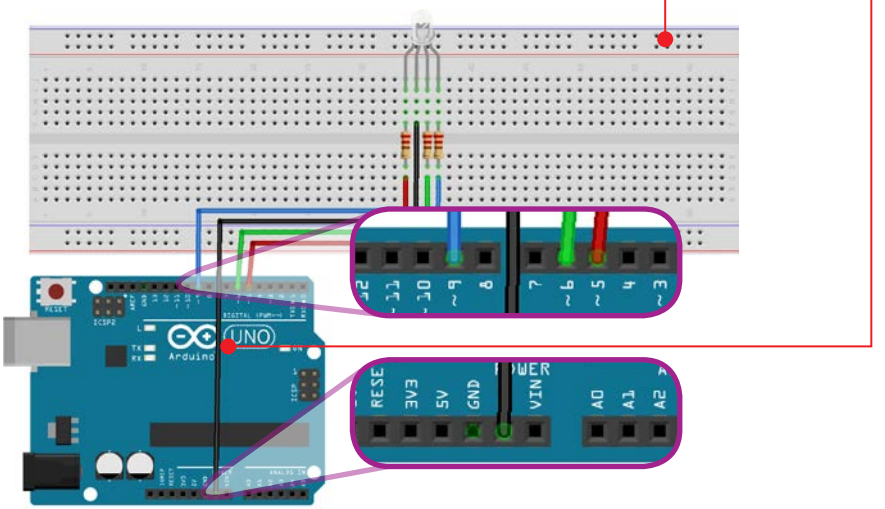
1 Rozpoczynamy budowanie obwodu. W przykładzie została użyta dioda LED ze wspólną katodą (masą), ale w sprzedaży są także diody RGB ze wspólną anodą (wtedy najdłuższą nóżkę należy podłączyć do 5V na płytce Arduino). Ustawimy przed sobą diodę tak, żeby najdłuższa nóżka była druga z lewej strony. W takim ustawieniu pierwsza z lewej będzie odpowiedzialna za kolor czerwony, trzecia za zielony, a czwarta za niebieski.

UWAGA!

Umiejętność odczytywania wartości rezystorów jest bardzo przydatna. Jeśli mamy opornik i chcemy wiedzieć, jaka jest jego wartość, możemy to sprawdzić na tej stronie: serwis-tv.com/opornik.html. Wystarczy ułożyć na niej taki sam zestaw kolorów, jak na naszym rezystorze.

2 Podłączamy czarnym kablem dłuższą nóżkę do GND na płytce Arduino ●.

3 Ustawiamy trzy rezystory 220 Ω i trzy przewody tak, by każdą z pozostałych nóżek podłączyć do wejść 5, 6 i 9 na płytce Arduino. Dla porządku ustalmy, że kolor czerwony podłączamy do wejścia 5, zielony do 6, a niebieski do 9. Cały obwód powinien być podłączony tak, jak ma to miejsce na schemacie ●.



4 Połączmy teraz kablem USB Arduino i nasz komputer. Musimy uruchomić program S4A i poczekać, aż zniknie komunikat mówiący o oczekiwaniu na podłączenie Arduino.

5 Teraz zaczniemy kodować. Nasz program będzie opierał się na odczytywaniu wciśnięć klawiszy na klawiaturze. Możemy ustawić ich działanie na przykład według następującego schematu:

- **klawisz** A – wyjście 5 – wartość 255 – włączy się kolor czerwony,
- **klawisz** Z – wyjście 5 – wartość 0 – kolor czerwony się wyłączy,
- **klawisz** S – wyjście 6 – wartość 255 – włączy się kolor zielony,
- **klawisz** X – wyjście 6 – wartość 0 – kolor zielony się wyłączy,
- **klawisz** D – wyjście 9 – wartość 255 – włączy się kolor niebieski,
- **klawisz** C – wyjście 9 – wartość 0 – kolor niebieski się wyłączy.

Dobór klawiszy jest dowolny, ten zaproponowany powyżej jest wygodny – wszystkie potrzebne klawisze znajdują się obok siebie na klawiaturze. Od teraz, wciśnięcie jednego z wymienionych klawiszy będzie powodowało włączanie się i gaszenie odpowiednich kolorów diody.

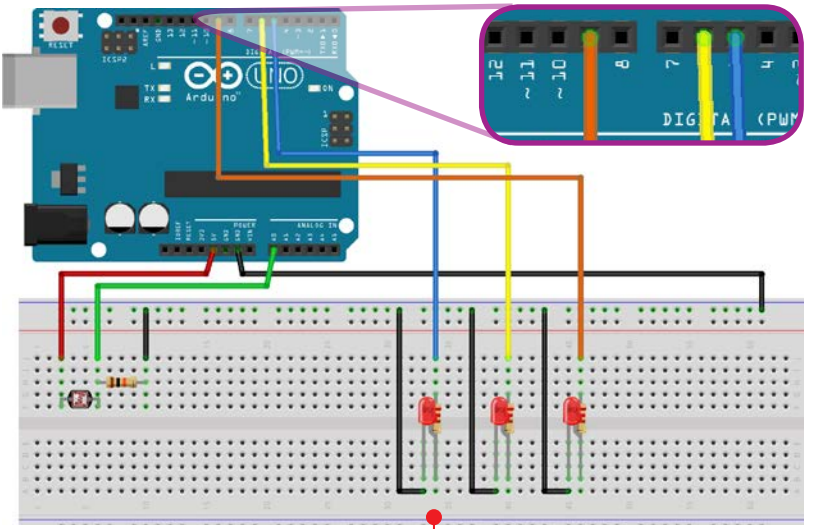
UWAGA!

Jeśli S4A przez dłuższy czas nie może nawiązać połączenia z Arduino, trzeba ponownie przejść przez proces instalacji firmware, opisany we wcześniejszym podrozdziale.

Budujemy czujnik oświetlenia

Kolejny projekt, który zbudujemy z Arduino, to czujnik natężenia światła. Wykorzystamy do niego informacje z poprzednich porad, dotyczące podłączania diod. Poznamy również działanie fotorezystora – czujnika, który potrafi mierzyć, jak jasno jest w otoczeniu. Będziemy potrzebowali:

- Arduino,
- płytki stykowej,
- przewodów,
- trzech diod LED dowolnego koloru
- trzech rezystorów 220 Ω,
- jednego rezystora 10 kΩ,
- fotorezystora.

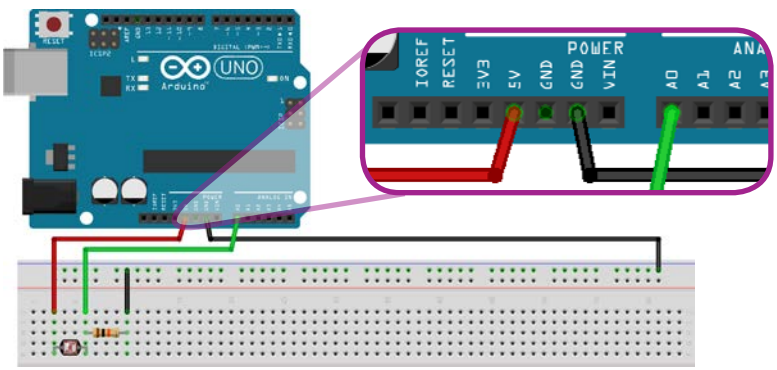


Nasze urządzenie będzie działało w ten sposób, że w zależności od tego, jak jasno jest w otoczeniu, będzie zapalała się określona liczba diod. Jeśli będzie ciemno – nie zapali się żadna, a jeśli bardzo jasno – wszystkie trzy. Stanom pośrednim będzie odpowiadało zapalenie jednej lub dwóch diod.

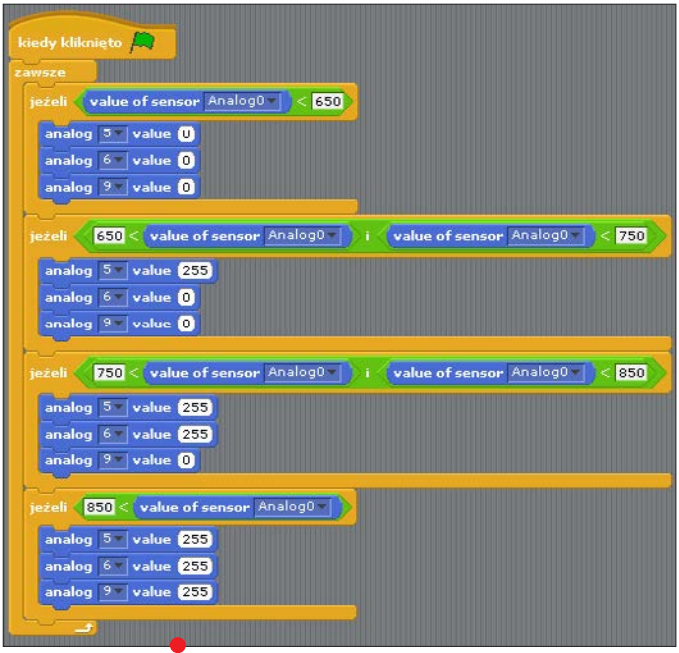
1 Przygotowujemy elementy potrzebne do zbudowania obwodu. Zaczniemy od podłączenia fotorezystora. Musimy doprowadzić do niego napięcie 5 V, a potem podłączyć go do jednego z wejść **Analog In** widocznych na płytce Arduino. Może to być na przykład A0. Równolegle dołączamy do obwodu rezystor 10 kΩ i dalej podłączamy obwód do GND – jak na schemacie ●.

2 Teraz podłączymy trzy diody LED. Robimy to tak, jak miało to miejsce w pierwszym z przykładów (patrz od strony 93). Możemy je podłączyć do wyjść 5, 6, 9. Pamiętajmy o dodaniu oporników 220 Ω, inaczej diody mogą się spalić. Połączenie całego zestawu powinno wyglądać jak na schemacie ●.

3 Uruchamiamy program S4A i łączymy płytkę Arduino kablem USB z naszym komputerem. Zauważmy, że po podłączeniu płytki wartość **Analog 0** utrzymuje się na sta-



łym poziomie, a inne bardzo szybko się zmieniają. To właśnie odczyt z fotorezystora – podłączyliśmy go do A0. Na podstawie odczytów tego wejścia ustawimy skrypt naszego małego urządzenia.

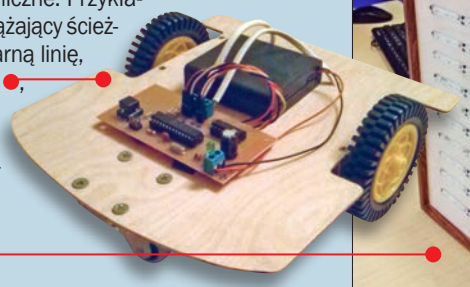


4 Stwórzmy jedną pętlę **zawsze**, która będzie uruchamiana po wciśnięciu zielonej flagi. Wewnątrz tej pętli będą znajdowały się cztery instrukcje warunkowe, które będą obsługiwały świecenie diod od trzech do jednej oraz możliwość nieświecenia żadnej z diod – gdy będzie naprawdę ciemno. Oto propozycja kodu. Po ułożeniu go wystarczy kliknąć na zieloną flagę, by zobaczyć, jak diody gasną i zapalają się, kiedy na przykład przykryjemy fotorezystor kartką lub poświecimy na niego latarką w telefonie komórkowym.

Fot.: Archiwum autora

ROBOTY I MIERNIKI

Trzy projekty przedstawiające połączenie Scratcha i Arduino pokazują, jak można zacząć eksperymentować. Arduino ma ogromną bazę kompatybilnych czujników mierzących natężenie głosu, wilgotność otoczenia, ruch i wiele innych. Jeśli poświęcilibyśmy więcej czasu na pracę z Arduino, moglibyśmy tworzyć całe urządzenia elektroniczne. Przykładem może być robot podążający ścieżką wyznaczoną przez czarną linię, nazywany line follower, albo miernik natężenia dźwięku wyświetlający aktualny pomiar w decybelach.



Scratch na Raspberry Pi

Kolejnym narzędziem, które doskonale nadaje się do współpracy ze Scratchem, jest Raspberry Pi. To komputer wielkości karty kredytowej. Jeśli tylko podłączymy do niego klawiaturę, mysz i monitor, będziemy mogli na nim normalnie pracować. Oczywiście nie jest to demon szybkości – nie to było jednak celem budowy Raspberry. „Malina" została specjalnie zaprojektowana, żeby uczyć programowania i elektroniki oraz sprawić, że będzie to bardzo angażujące zajęcie. Koszt takiego komputera to około 200 złotych.

Świat oszalał na punkcie rozmaitości wynalazków, jakie można stworzyć z wykorzystaniem „Maliny". Można dzięki niej zbudować:

- konsole dla starych gier,
- urządzenia do karmienia zwierząt,
- systemy kontroli dla inteligentnego domu,
- kioski informacyjne,
- narzędzia pomiarowe środowiska w stratosferze,
- aparaty fotograficzne,
- smartfony,
- stacje pogodowe.

Komputer Raspberry Pi podłączony i działający

Komputer do Scratcha

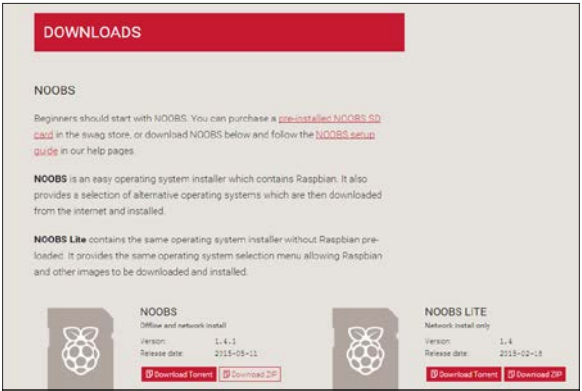
Możemy zacząć zabawę z Raspberry i Scratchem. Zbudujemy komputer do programowania w Scratchu. Będziemy potrzebowali:

- komputer Raspberry Pi,
- kartę SD o pojemności przynajmniej 4 GB (będzie pełniła funkcję dysku twardego),
- kabel HDMI,
- zasilacz (może wystarczyć zwykła ładowarka do telefonu),
- mysz i klawiaturę (obie na USB),
- monitor ze złączem HDMI (może to też być telewizor),
- inny komputer i czytnik kart SD.

1 By móc korzystać z Raspberry Pi, musimy najpierw przygotować system operacyjny. Nasz mały komputer będzie działał na specjalnie przygotowanej wersji systemu Linux – **Raspbian**. Zainstalujemy go za pomocą instalatora o nazwie **Noobs**. Znajdziemy go w KŚ+ lub pobierzemy ze strony **raspberrypi.org/downloads** (plik jest dość duży – około 750 MB – decydując się na pobieranie, musimy się więc uzbroić w cierpliwość).

2 W tym czasie powinniśmy pobrać program do formatowania kart SD. Znaj-

dziemy go na płycie dołączonej do książki lub na stronie **sdcard.org** w dziale **Downloads**. Pobrany plik rozpakowujemy i instalujemy.



Strona, z której możemy pobrać Noobs dla Raspberry Pi

3 Po pobraniu Noobsa rozpakowujemy archiwum, a jego zawartość przenosimy na kartę SD. Gdy ten proces dobiegnie końca, bezpiecznie usuwamy kartę z komputera i wkładamy ją do Raspberry.

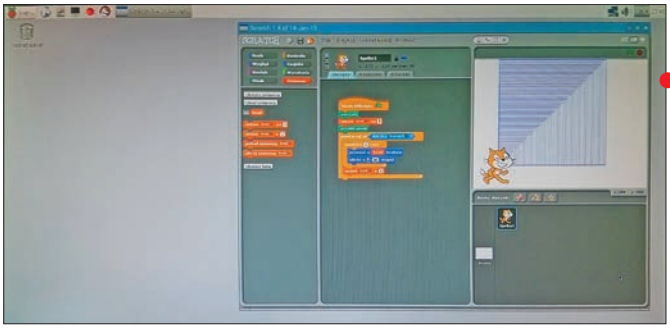
4 Podłączamy wszystkie podzespoły do „Maliny" – mysz, klawiaturę, monitor i na końcu zasilacz, to on sprawi, że komputer się uruchomi.

5 Przechodzimy teraz do pierwszej konfiguracji systemu operacyjnego. Nasze Raspberry chwilę po podłączeniu źródła prądu wyświetli okno z listą możliwych do zainstalowania systemów operacyjnych. Najbardziej polecanym jest **Raspbian** – to właśnie jego wybierzmy. System zacznie się instalować. Ta operacja może zająć chwilę.

6 Gdy proces instalacyjny się zakończy, pojawi się na monitorze tekstowe menu konfiguracyjne. Możemy zmienić datę i godzinę, a także wiele innych przydatnych ustawień. Gdy skończymy, musimy przejść na pole **Finish** za pomocą klawisza tab.

7 System może nie robi wielkiego wrażenia – to dlatego, że standardowo uruchamia się w trybie tekstowym. Zaraz jednak to zmienimy. Logujemy się na nasze konto. Domyślny login to **pi**, a hasło to **raspberry**. Po zatwierdzeniu tych danych wpisujemy w konsoli polecenie **startx**, a system za chwilę włączy tryb graficzny.

Menu konfiguracyjne naszego małego komputera

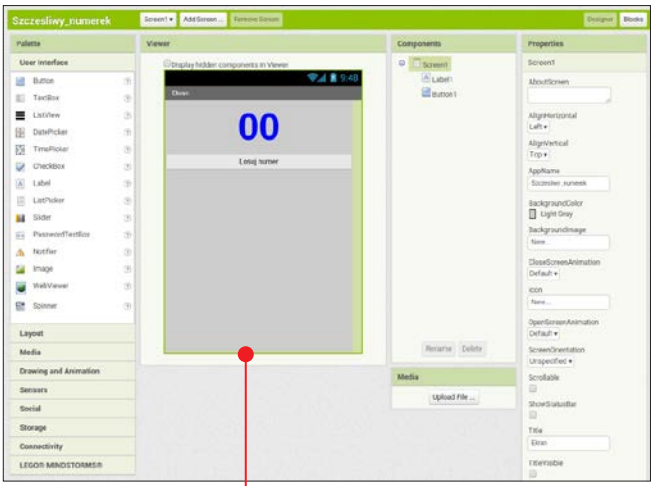


instalowanych programów dla Raspberry. Udało nam się więc zbudować od zera minikomputer do scratchowania. Jeśli chcielibyśmy się nim pochwalić innemu użytkownikowi mającemu Raspberry, to wystarczy, że przeniesiemy do niego naszą kartę SD. To doskonałe rozwiązanie na przykład dla nauczycieli i uczniów pracujących z „malinowym komputerem".

Możemy już zacząć zabawę z naszym komputerkiem. Z pewnością na pierwszym planie zobaczymy ikonę Scratcha – to jeden z pre-

AppInventor – aplikacje mobilne

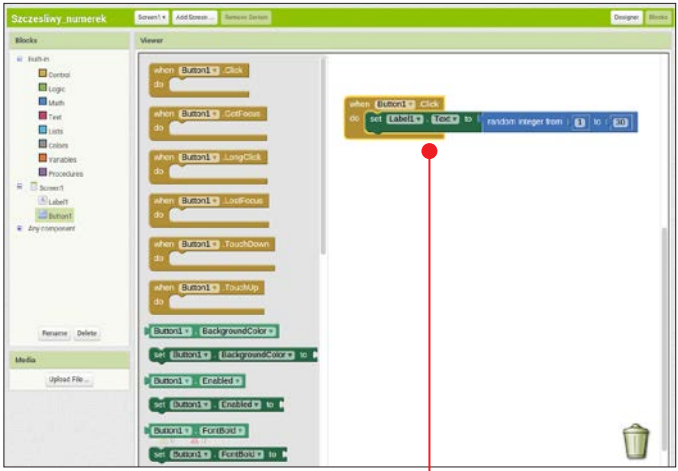
Być może przyjdzie taki moment, że zechcemy tworzyć aplikacje mobilne. Nie jest to trudne, nawet jeżeli jedynym środowiskiem programistycznym, które znamy, jest Scratch. Najważniejsza jest przecież sama umiejętność programistycznego myślenia, a nie środowisko. Tak się akurat składa, że dla Androida – czyli najpopularniejszego mobilnego systemu operacyjnego na świecie – istnieje bardzo proste i podobne do Scratcha środowisko programistyczne. Nazywa się **AppInventor** i jest przygotowane przez specjalistów z MIT.



Sporo informacji o tym środowisku znajdziemy na stronie **appinventor.mit.edu** (na tej stronie można między innymi dowiedzieć się, jak skonfigurować telefon, by testować na nim nasze aplikacje).

Jak stworzyć aplikację mobilną

Zobaczmy, jak stworzyć aplikację mobilną w AppInventorze – niech to będzie dla przykładu program losujący liczby w zakresie od 1 do 30. Świetnie sprawdzi się choćby do wyznaczania „chętnych" uczniów do odpowiedzi.

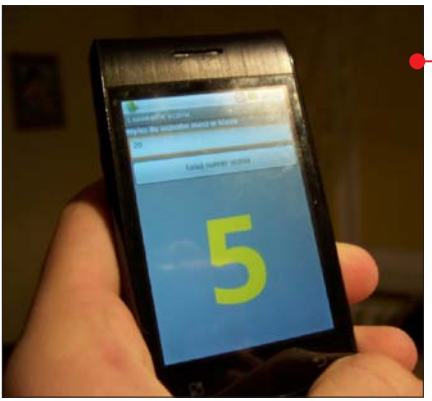


1 Samo środowisko programistyczne znajdziemy na stronie **ai2.appinventor.mit.edu**. Pracę musimy rozpocząć od stworzenia nowego projektu. Nazwijmy go **Szczesliwy_numerek**. W środku widzimy ekran aplikacji, która na razie jest pusta. Dodajmy element o nazwie **Label**, a pod nim **Button**. Po kliknięciu na każdy z tych obiektów po prawej stronie ukażą się jego parametry (**Properties**). Poświęćmy im chwilę i samodzielnie postarajmy się je tak pozmieniać, aby uzyskać podobny efekt.

2 Skończyliśmy projektować interfejs graficzny naszego programu losującego. Nie jest on może bardzo efektowny, ale swoją funkcję będzie spełniał. Przejdźmy teraz do kodowania. Ekran tworzenia skryptów włączymy, klikając na **Blocks** w prawym górnym rogu okna. Co się powinno dziać? Otóż po naciśnięciu przycisku z napisem **Losuj numer** zawartość obiektu **Label** ma być ustawiana na losową wartość od 1 do 30. Tak też ułóżmy odpowiednie bloczki. Zauważmy, że AppInventor jest bardzo podobny do Scratcha.

To już koniec. Program jest gotowy. Teraz możemy go przetestować na naszym smartfonie. Możemy w podobny sposób tworzyć inne aplikacje wykorzystujące możliwości telefonu. Nasze programy mogą używać aparatu fotograficznego, nagrywać dźwięk, korzystać z GPS, pisać e-maile i robić wiele, wiele innych rzeczy.

Gdy programujemy, nasze możliwości są ograniczone jedynie naszą wyobraźnią!



Biblioteczka Komputer Świat

AUTOR: Piotr Szlagor

REDAKTORZY PROWADZĄCY: Paweł Paczuski, Rafał Kamiński, Agnieszka Al-Jawahiri

PRZYGOTOWANIE PŁYTY: Mariusz Michalski

PROJEKT OKŁADKI: Robert Dobrzyński

DZIAŁ DTP: Robert Dobrzyński, Mariusz Rybak

KOREKTA: Jolanta Rososińska

ringier axel springer

ISBN: 978-83-7813-929-4
Warszawa 2015

Wydawca: RINGIER AXEL SPRINGER POLSKA Sp. z o.o.
02-672 Warszawa, ul. Domaniewska 52
tel. 22 2320000, 22 2320001
www.ringieraxelspringer.pl

DYREKTOR ZARZĄDZAJĄCY MARKĄ: Paweł Paczuski

BUSINESS PROJECT MANAGER: Paweł Bulwan

DRUK I OPRAWA: Toruńskie Zakłady Graficzne ZAPOLEX Sp z o.o.

EGZEMPLARZE ARCHIWALNE:
tel. 22 3367901
infolinia 801 000869
prenumerata.axel@qg.com (zamówienia)

SPRZEDAŻ INTERNETOWA: www.literia.pl

KONTAKT:
TEL: 22 2320078 (w godzinach 11–15)

INTERNET: www.komputerswiat.pl, ksplus.pl

E-MAIL: biblioteczka@komputerswiat.pl

KOMPUTER
ŚWIAT
BIBLIOTECZKA

NAUKA PROGRAMOWANIA PRZEZ ZABAWĘ

ABC PROGRAMOWANIA

Scratch to środowisko do nauki programowania, a ta książka to podręcznik do niego. Nauczymy się z niej, jak tworzyć kształty z wykorzystaniem funkcji rysowania w Scratchu. Rysując, zobaczymy, jak działa pętla. Potem przeczytamy, jak korzystać ze zmiennych, list i instrukcji warunkowych. Zakodujemy proste i trudniejsze programy, a potem przejdziemy do tworzenia gier.

Ostatni rozdział – to inspiracja na przyszłość. Znajdziemy w nim informacje, jak – znając już podstawy programowania – zacząć przygodę z tworzeniem urządzeń elektrycznych i aplikacji na smartfony.

W KŚ+ (www.ksplus.pl) znajdziemy 30 dodatkowych stron – wzory rysunków do przeglądania, pobrania i wydrukowania w formacie PDF, wraz z podpowiedziami, rozwiązaniami i gotowymi skryptami, oraz zadania sprawdzające wiedzę ze skryptami rozwiązań.

CENA 14,90 zł
w tym 5% VAT

ISBN 978-83-7813-929-4 INDEKS 321 958 03
9 788378 139294

Nr 3/2015 (83)

KOMPUTER ŚWIAT BIBLIOTECZKA