

SCRATCH: PODSTAWY NA PRZYKŁADZIE GRY TYPU PING-PONG

Marek Nowicki
ul. Chopina 12/18, 87-100 Toruń
Wydział Matematyki i Informatyki
Uniwersytet Mikołaja Kopernika w Toruniu
m.nowicki@mat.umk.pl

Abstract. *This paper presents the fundamentals of Scratch programming. In this paper, I will demonstrate how to create a ping-pong-style game from scratch in Scratch, where two players control paddles and bounce the ball. The goal is to bounce the ball so that it lands off the board on the opponent's side. The game's graphics will be simple: a pre-made ball element will be used, and the paddles will be made as rectangular costumes, so students will not spend too much time drawing and their attention will be focused primarily on programming.*

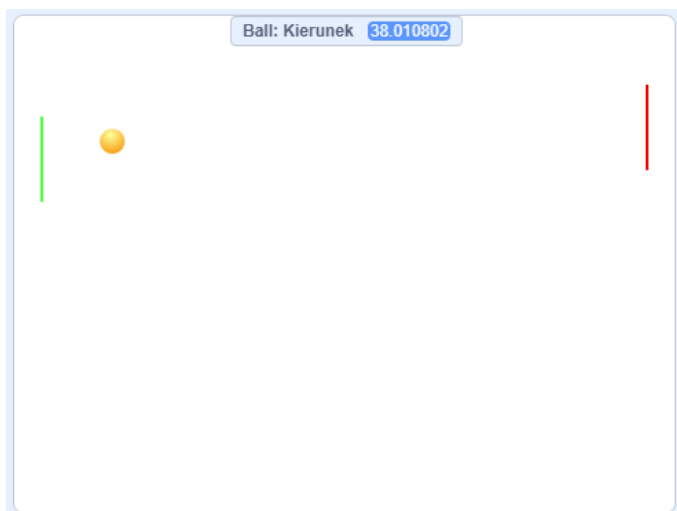
1. Wstęp

Język programowania Scratch [1] (wym. *skracz*, [*skrætf*]) został zaprojektowany z myślą o dzieciach i młodzieży w wieku szkolnym. Ułatwia on naukę podstaw programowania, ponieważ nie wymaga znajomości składni języka ani wpisywania tekstowych poleceń. Zamiast tego wykorzystuje wizualne podejście do tworzenia programów, w którym instrukcje są reprezentowane przez kolorowe bloczki.

W ramach wcześniejszych edycji konferencji Informatyka w Edukacji, opublikowane były artykuły m.in. ze wstępem do nauki programowania w języku Scratch [3], o wykorzystaniu języka Scratch do nauki programowania w szkole podstawowej [2], możliwości wykorzystania go do przedstawienia idei programowania w przedszkolu [6]. Pojawiały się prace przedstawiające nowości w Scratchu – czy to wprowadzające do Scratcha w wersji 3.0 [4], czy to do rozszerzeń dla środowiska Scratch [5]. Dodatkowo, artykuł [7] pokazywał podstawy programowania w języku Scratch na przykładzie gry w węża (*snake*).

W niniejszej pracy zaprezentuję, jak od podstaw stworzyć grę typu ping-pong, w której dwóch graczy steruje paletkami i odbija piłeczkę. Celem gry jest takie odbijanie piłeczki, aby wypadła ona poza planszę po stronie przeciwnika. Warstwa graficzna gry

będzie prosta: wykorzystany zostanie gotowy element reprezentujący piłkę, natomiast paletki zostaną przygotowane samodzielnie jako prostokątne kostiumy. Dzięki temu uczniowie nie będą poświęcać zbyt dużo czasu zajęć na rysowanie, a ich uwaga będzie skupiona przede wszystkim na programowaniu. Rysunek 3 przedstawia przykładowy wygląd gry.



Rysunek 1. Przykładowy wygląd gry

2. Podstawy

W niniejszej pracy używana będzie najnowsza wersja *online* (przeglądarkowa) środowiska Scratch bazująca na wersji 3.0, której pierwsze wydanie miało miejsce ponad 6 lat temu, bo w 2019 roku. Dostępne są również wersje *offline* (samodzielne aplikacje, które nie wymagają do pracy połączenia z Internetem) środowiska Scratch na systemy Windows 10+, macOS 10.13+, ChromeOS i Android 6.0+ (w tym przypadku działająca jedynie na tabletach a nie telefonach), ale ich interfejsy są zbliżone do wersji *online*, w związku z tym nie będą osobno omawiane.

Aby rozpocząć przygodę ze środowiskiem Scratch, po wejściu na jego stronę główną (<https://scratch.mit.edu/>) należy wybrać przycisk *Stwórz* (*Create*), który przeniesie użytkownika na stronę edytora projektu (<https://scratch.mit.edu/projects/editor/>).

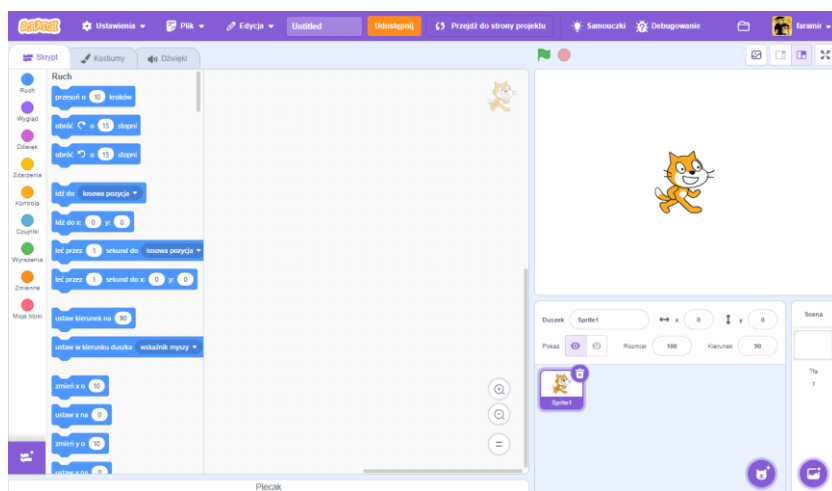
2.1. Edytor projektu

Rysunek 2 przedstawia główny widok edytora Scratch. Od wersji 1.4 środowiska Scratch, edytor posiada polską wersję językową. Na górze edytora znajduje się pasek menu, który umożliwia m.in. zmianę języka interfejsu użytkownika. Jeśli edytor zostanie

uruchomiony w angielskiej wersji językowej, można wybrać język polski korzystając z opcji: ⚙ *Settings* > 🌐 *Language* > *Polski*.

Menu pozwala również m.in. na zapisywanie projektu na serwerze Scratch (po zalogowaniu się na konto zarejestrowanego użytkownika) oraz na lokalnym komputerze, a także na wczytywanie projektów zapisanych na komputerze.

Poniżej paska menu znajduje się główna część edytora. Po lewej stronie umieszczony jest panel zawierający zakładki: *Skrypt*, *Kostiumy/Tła* i *Dźwięki*. Po prawej stronie znajdują się dwa kolejne panele: u góry widok sceny, czyli obszar widoczny dla użytkowników stworzonej aplikacji, w którym będą wykonywane zaprogramowane akcje, a poniżej panel zawierający wszystkie duszki użyte w projekcie oraz ich właściwości, m.in. nazwę, położenie i kierunek, w który skierowany jest duszek. W panelu tym znajduje się również element oznaczony nazwą *Scena*, związany z tłem aplikacji, dźwiękami



Rysunek 2. Główny widok edytora projektu

i akcjami sceny, czyli globalnymi, niezależnymi od konkretnego duszka.

Podświetlony element (na rysunku *Sprite1*) oznacza obiekt, który będzie modyfikowany i do którego odnoszą się skrypty, kostiumy oraz dźwięki widoczne w lewej części edytora. Ważne jest, aby w trakcie pracy ze Scratchem zaznaczony był odpowiedni obiekt, duszek lub scena, na którym aktualnie pracujemy.

2.2. Bloczki, czyli instrukcje

Programy w Scratchu tworzy się przez przeciąganie bloczków, czyli instrukcji, z zasobnika znajdującego się po lewej stronie edytora do obszaru roboczego w środkowej

części okna. Poszczególne kategorie (grupy) instrukcji oznaczone są różnymi kolorami: na przykład bloczki z kategorii Ruch są niebieskie, Wygląd – fioletowe, a Dźwięk – różowe. Kształt bloczków wskazuje, w jaki sposób można je ze sobą łączyć – wcięcia i wypustki pełnią podobną funkcję jak elementy puzzli. Niektóre instrukcje zawierają także pola, w których można umieszczać wartości, tekst lub inne bloczki pełniące rolę parametrów. Rozłączanie instrukcji odbywa się w podobny sposób co łączenie, czyli przez przeciągnięcie wybranego bloczka poza utworzony układ. Bloczek można usunąć, przenosząc go z powrotem do zasobnika albo wybierając opcję *Usuń blok* z menu kontekstowego dostępnego po kliknięciu bloczka prawym przyciskiem myszy.

Niektóre bloczki nie mają wycięcia w górnej części, lecz zaczynają się łukiem. Są to bloczki początkowe, od których rozpoczyna się wykonywanie skryptu. Najważniejszym z nich jest instrukcja `kiedy kliknięto [zieloną flagę]`, ponieważ umożliwia ona skoordynowane uruchomienie stworzonej aplikacji po naciśnięciu zielonej flagi znajdującej się nad widokiem sceny.

Część bloczków ma kształt prostokąta z zaokrąglonymi bokami. Są to bloczki związane ze zmiennymi oraz operacjami zwracającymi wartości liczbowe lub tekstowe. Można je umieszczać w odpowiednich polach innych instrukcji. Podobną rolę pełnią bloczki sześciokątne, które reprezentują operacje zwracające wartości logiczne.

Bloczki z grupy Kontrola obejmują instrukcje sterujące przebiegiem skryptu. Wśród nich są instrukcje pętli, takie jak: `zawsze` (pętla nieskończona), `powtarzaj aż (...)` (pętla warunkowa), czy `powtórz (...) razy` (pętla powtórzeniowa) oraz instrukcje warunkowe `jeżeli <...> to ... w przeciwnym razie ...`. Instrukcje te pozwalają na umieszczanie innych bloczków wewnątrz ich struktury, dlatego bywają potocznie nazywane *krokodylkami*. W tej samej grupie znajdują się także instrukcje końcowe, które nie mają dolnej wypustki, lecz są zakończone prostą krawędzią, np. `zawsze` oraz `zatrzymaj [wszystko]`. Za takimi bloczkami nie można dodawać kolejnych instrukcji.

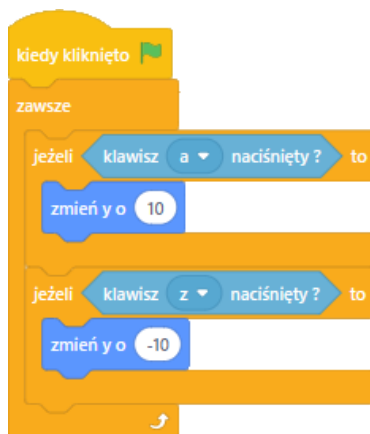
3. Gra typu ping-pong

Gra w ping-ponga może być bardzo rozbudowana, ale przedstawiona wersja będzie ograniczona jedynie do ruchu piłeczki po planszy i sterowania paletkami dwóch graczy. Aby rozgrywka była trochę mniej przewidywalna, odbicia paletkami nie będą zachowywały na całej swojej długości właściwości, że kąt padania równa się kątowi odbicia, ale kąt będzie się zwiększał wraz ze zwiększeniem odległości od środka paletki. Oczywiście gra będzie kończyć się, gdy piłeczka dotknie prawej lub lewej krawędzi sceny.

3.1. Paletki

Jako, że duszek przedstawiający kota nie będzie potrzebny, należy go usunąć klikając na ikonę kosza w przestrzeni dostępnych duszków i potwierdzić usuwanie.

Następnie stworzymy duszka – pierwszą paletkę. W tym celu najeżdżamy na ikonę kota z plusem, znajdującą się w prawym dolnym rogu zasobnika z duszkami, i wybieramy ikonę pędzla. Zostanie otwarta zakładka *Kostiumy*, aby namalować pierwszy kostium – wizerunek duszka. Za pomocą przycisku z lupą z plusem wykonujemy przybliżenie ok. 10-12 krotnie, by narysować paletkę – poziomą linię w dowolnym kolorze, np. zielonym i ustawiamy ją na środek obszaru, co jest reprezentowane przez celownik. Możemy od razu nazwać duszka jako *paletka1*, obrócić duszka za pomocą własności *kierunek* z domyślnych 90° na 0° i ustawić współrzędną *x* na -220, by paletka była ustawiona pionowo.



Rysunek 3. Pierwszy skrypt paletki



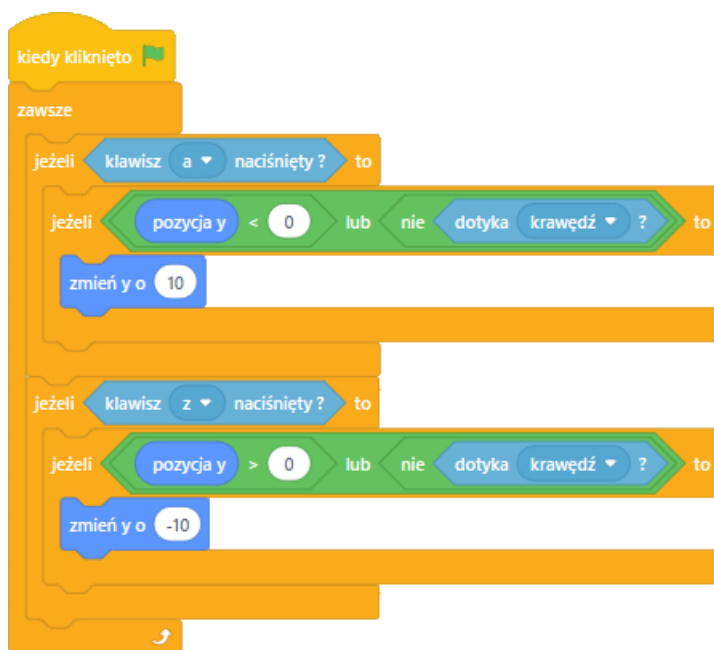
Rysunek 4. Drugi skrypt paletki

Gdy mamy gotowy kostium, to przechodzimy do zakładki *Skrypty*. Ze względu na to, że chcemy by program reagował na naciśnięcie przycisków obu graczy, nie możemy skorzystać z bloczku *Kiedy klawisz [spacja] naciśnięty*, bo w trakcie przytrzymania klawisza tylko jedna z akcji będzie mogła być wykonywana. Zamiast tego zastosujemy sondowanie naciśnięcia klawisza.

W tym celu na początek wybierzemy bloczek startowy *kiedy kliknięto [zieloną flagę]* (grupa *Zdarzenia*) i dolożymy do niego instrukcję pętli *zawsze* (grupa *Kontrola*). Wewnątrz tej konstrukcji będziemy sprawdzali, czy odpowiednie klawisze są wciśnięte korzystając z bloczka *klawisz [spacja] naciśnięty?* (grupa *Czujniki*), a jeśli tak, to będziemy zmieniać wartość współrzędnej *y* za pomocą bloczka *zmień y o (10)* (grupa *Ruch*). Przykładowy skrypt poruszający paletką w górę i w dół za pomocą klawiszy *a* i *z* przedstawia Rysunek 3.

W tej wersji skryptu widzimy, że paletka może prawie cała uciec poza planszę. Możemy ten błąd poprawić przez sprawdzanie, czy po ruchu duszek dotyka krawędzi za pomocą odpowiedniego bloczka z grupy *Czujniki* i jeśli tak, to możemy cofnąć ostatni ruch. Zmianę tę przedstawia Rysunek 4.

Przy zastosowaniu powyższego sposobu poradzenia sobie z powyższym problemem, paletka nie wyjdzie za ekran, ale również nigdy nie pozwoli na przysunięcie się do brzegu sceny. Aby pozwolić na delikatne najechanie na krawędź sceny można zezwolić na przesunięcie, tylko wówczas, gdy poruszając się do góry, *pozycja y* (grupa *Ruch*) jest mniejsza od 0 (grupa *Wyrażenia*) lub duszek paletki nie dotyka krawędzi (grupa *Czujniki*). Analogicznie dla poruszania się w dół. Przykład skryptu implementującego to zachowanie przedstawiona Rysunek 5.



Rysunek 5. Ostateczny skrypt paletki

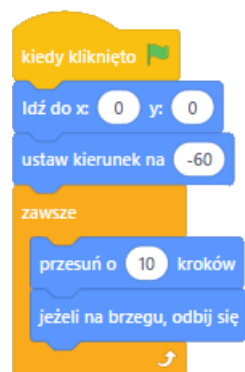
Analogicznie będziemy tworzyć duszka drugiej paletki – drugiego gracza. Można wykonać te same czynności jeszcze raz albo kliknąć na już stworzonego duszka prawym przyciskiem myszy i wybrać polecenie *duplikuj*. Warto wówczas zmienić kolor kostiumu, np. na czerwony, wybierając na zakładce *Kostiumy* linię przedstawiającą pa-

letkę i zmienić kolor zarysu. Dodatkowo należy zmienić współrzędną x na 220, by paletka była po prawej stronie planszy. Dodatkową zmianą jest zmiana klawiszy z a i z na strzałka w górę i strzałka w dół.

3.2. Piłeczka

Kolejnym duszkiem będzie piłeczka. Wybierzemy ją z gotowych duszków przez ikonę koła z plusem w zasobniku duszków. Wśród dostępnych predefiniowanych duszków znajdujemy duszka nazwanego *Ball*. We właściwościach zmniejszamy jego rozmiar do 40%. W zakładce *Kostiumy* można wybrać jeden z pięciu kolorów kostiumu duszka.

Gdy kostium duszka jest już przygotowany, przejdziemy do zakładki *Skrypty*. Na początek w grupie *Ruch* zaznaczamy pole wyboru obok bloczka *kierunek*, aby na scenie pojawiła się informacja o aktualnym kierunku ruchu piłeczki.

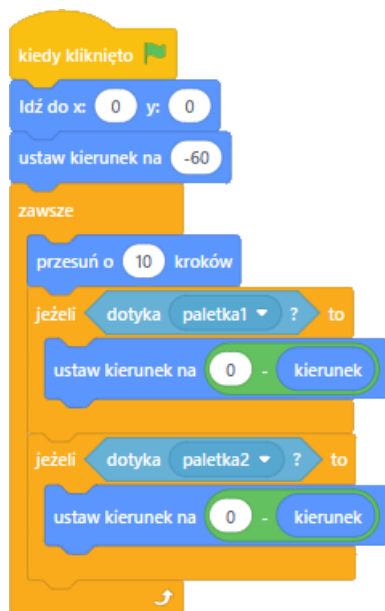


Rysunek 6. Pierwszy skrypt piłeczki

Następnie zaczynamy opisywać zachowanie piłeczki. Rozpoczynamy od bloczku startowego *kiedy kliknięto [zieloną flagę]*, po którym ustawimy współrzędne za pomocą instrukcji *idź do x: (0) y: (0)* (grupa *Ruch*) oraz *ustaw kierunek na (-60)*. Następnie w pętli *zawsze* dodamy instrukcję *przesuń o (10) kroków*, czym spowodujemy ruch piłeczki. Po uruchomieniu programu zieloną flagą piłeczka przesuwa się do krawędzi, ale nie odbija się. Można samodzielnie sprawdzić, czy piłka dotyka krawędzi i ustawić nowy kierunek ruchu obliczając kąt po odbiciu od krawędzi górnej i dolnej (dokładnie $(180) - (\text{kierunek})$), ale dużo prościej jest zastosować bloczek *jeżeli na brzegu, odbij się*. Rysunek 6 przedstawia przykładowy skrypt.

Teraz widać, że piłeczka porusza się po całej planszy i odbija od krawędzi sceny, ale nie reaguje na paletki. Jest to spowodowane tym, że nie zaprogramowaliśmy jeszcze reagowania na to zdarzenie. W tym celu przed bloczkiem *jeżeli na brzegu, odbij się* dodamy sprawdzenie, czy aktualny duszek dotyka duszka *paletka1*, a jeśli tak, to ustawimy dalszy kierunek ruchu na $(0) - (\text{kierunek})$. Skorzystamy z wyrażenia warunkowego *jeżeli <...> to* oraz czujnika *dotyka [paletka1 ▽]?*. Analogicznie postępujemy z reakcją na dotykanie duszka *paletka2*. Zmianę tę przedstawia Rysunek 7.

Zmiana ta pozwoliła, by paletki również odbijały piłkę. Niestety odbicia zawsze są takie same, bardzo przewidywalne. Zmienimy teraz sposób obliczenia kąta w taki sposób, by kąt był zależny od miejsca uderzenia paletki – im dalej od środka, tym większy

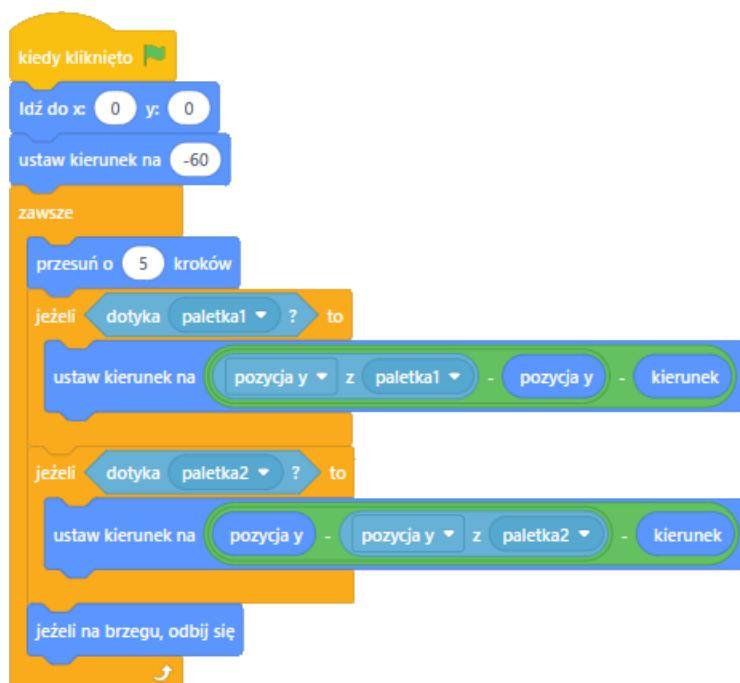


Rysunek 7. Drugi skrypt piłeczki

kąt odbicia. Scratch udostępnia w grupie *Czujniki* bloczek zwracający odległość od innych duszków czy wskaźnika myszy, ale jest to odległość absolutna, a my potrzebujemy informacji, czy odbicie miało miejsce w górnej czy dolnej części paletki. W tym celu zastosujemy bloczek, który domyślnie ma pola `[indeks tła ▾] z [scena ▾]`. Będziemy wyliczać odległość *pozycji y*, ale ten element menu dostępny będzie dopiero, gdy zamiast *scena* wybierzemy *paletka1* w tym bloczku. W miejsce wartości 0 przy bloczku *ustaw kierunek*, wstawimy wyrażenie: `[[pozycja y ▾] z [paletka1 ▾]] - [pozycja y]`. Pozwoli to na zmianę kąta odbicia. W przypadku *paletki2* kolejność elementów w wyrażeniu zamieniamy miejscami i powstaje `[[pozycja y] - [[pozycja y ▾] z [paletka2 ▾]]]`. Skrypt z omawianymi zmianami przedstawia Rysunek 8.

Do dokończenia pierwszej wersji gry pozostało zweryfikowanie wygranej po dotarciu piłeczki do krawędzi przeciwnika. Można to zrobić tworząc niewidoczne duszki (z ustawionym efektem *duch* na 100%). Duszki te mogą być po prostu długą linią ustawioną na krawędzi planszy z lewej i prawej strony.

Innym sposobem sprawdzania wygranej jest kontrolowanie, czy piłeczka dotyka krawędzi sceny. Jeśli jej pozycja na osi x jest mniejsza niż -230, wygrywa gracz po prawej



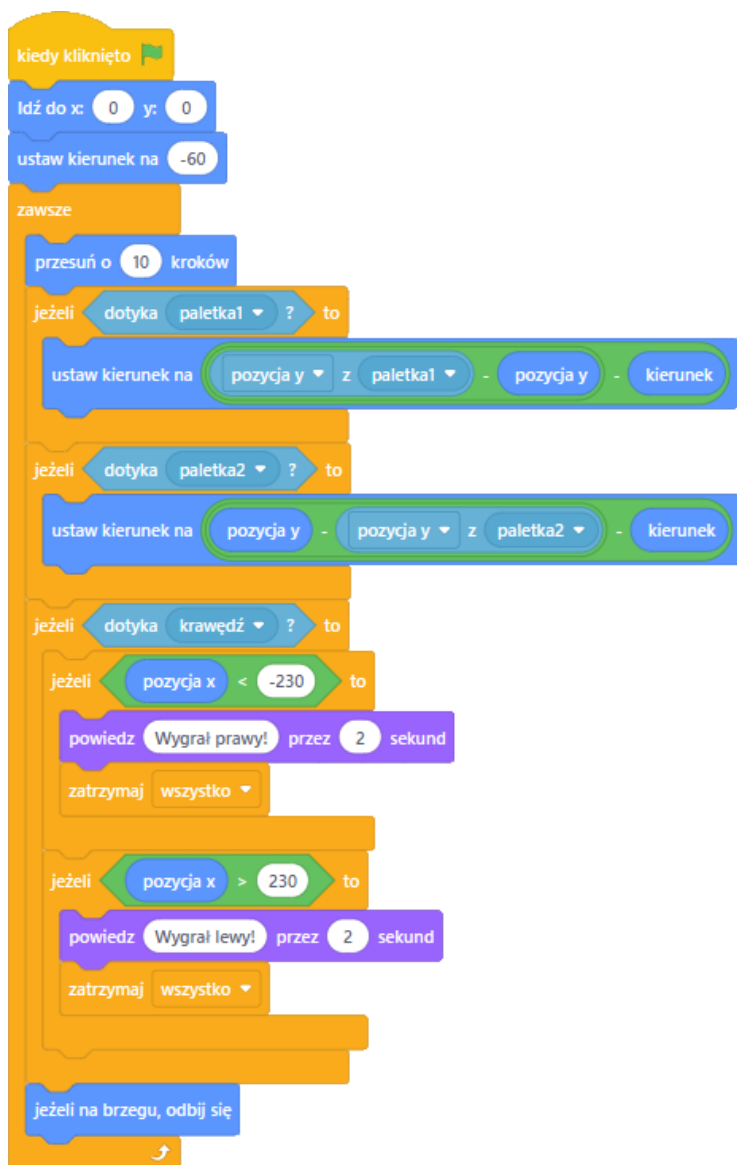
Rysunek 8. Trzeci skrypt piłeczki

stronie, a jeśli jest większa niż 230, wygrywa gracz po lewej stronie. Po spełnieniu jednego z tych warunków gra zostaje zatrzymana. Ze względu na prostotę wybrano tę wersję. Pełen, ostateczny skrypt piłeczki przedstawia Rysunek 9.

4. Podsumowanie

W niniejszej pracy przedstawione zostały podstawowe elementy środowiska Scratch oraz pokazany został sposób przygotowania prostej gry typu ping-pong. Zaprezentowany projekt nie jest skomplikowany, ale pozwala uzyskać w pełni działającą grę. Dzięki temu może stanowić dobry punkt wyjścia do dalszej pracy z uczniami, którzy po wykonaniu podstawowej wersji mogą rozszerzać ją o kolejne funkcje, na przykład licznik punktów, przeszkody pojawiające się na planszy albo bonusy wpływające na sposób poruszania się piłeczki lub wielkość pałek graczy.

Projekt można rozwijać również przez zmianę sposobu sterowania pałkami. W tym celu warto sprawdzić możliwość wykorzystania dostępnych w Scratchu rozszerzeń, takich jak bloczki związane z czujnikiem wideo (ang. *video sensing*) lub *Face Sensing Blocks*, które umożliwiają wykrywanie twarzy oraz jej wybranych elementów, m.in. oczu,



Rysunek 9. Ostateczny skrypt piłeczki

uszu, ust i nosa. Ciekawym uzupełnieniem projektu może być także użycie platformy *micro:bit* albo zestawów LEGO współpracujących ze Scratchem, które pozwoliłyby sterować paletkami za pomocą fizycznych urządzeń innych niż klawiatura komputera.

Dalszy rozwój gry może przebiegać w wielu kierunkach i zależy przede wszystkim od pomysłowości uczniów oraz ich gotowości do samodzielnego eksperymentowania z możliwościami Scratcha.

Literatura

1. *Scratch – Imagine, Program, Share*, <https://scratch.mit.edu>, ostatni dostęp 10.06.2026 roku
2. Bała L., Bała P., *Nauka programowania w szkole podstawowej : Scratch*, Informatyka w edukacji : kształcenie informatyczne i programowanie dla wszystkich uczniów / red. Anna Beata Kwiatkowska, Maciej M. Sysło, Toruń, Wydawnictwo Naukowe Uniwersytetu Mikołaja Kopernika, 2016. s. 94-104, ISBN 978-83-231-3585-2
3. Bała P., Nowicki M., *Scratch - środowisko do nauki programowania, Informatyka w edukacji, materiały z warsztatów VII Konferencji IwE 2010* / red. Anna Beata Kwiatkowska, Toruń, Wydawnictwo Naukowe Uniwersytetu Mikołaja Kopernika, 2011, s. 33-40, ISBN 978-83-231-2677-5
4. Bała P., Nowicki M., *Scratch 3.0: nowe możliwości, Informatyka w edukacji, myśl komputacyjnie!* / red. Anna Beata Kwiatkowska, Maciej M. Sysło, Toruń, Wydawnictwo Naukowe Uniwersytetu Mikołaja Kopernika, 2018, s. 173-174, ISBN 978-83-231-4049-8
5. Krowiak B., *Wykorzystanie dodatkowych rozszerzeń w środowisku Scratch*, / *Informatyka w edukacji: uczniowie i nauczyciele w szkole przyszłości* / red. Anna Beata Kwiatkowska, Maciej M. Sysło, Toruń, Wydawnictwo Adam Marszałek, 2022, s. 400-405, ISBN 978-83-8180-645-9
6. Nowicki M., *Scratch w przedszkolu, Informatyka w edukacji: uczniowie i nauczyciele w szkole przyszłości* / red. Anna Beata Kwiatkowska, Maciej M. Sysło, Toruń, Wydawnictwo Adam Marszałek, 2022, s. 423-434, ISBN 978-83-8180-645-9
7. Nowicki M., *Scratch : podstawy na przykładzie gry Wąż, Informatyka w edukacji : kształcenie informatyczne drogą do cyfrowej transformacji społeczeństwa* / red. Anna Beata Kwiatkowska, Toruń, Wydawnictwo Adam Marszałek, 2024, s. 284-294, ISBN 978-83-8180-904-7