Szczecińskie Collegium Informatyczne
Szczecin – ul. Mazowiecka 13
www.sci.edu.pl

SZCZECIŃSKIE COLLEGIUM INFORMATYCZNE

www.acodemia.pl **e-mail:** info@acodemia.pl

Lekcja 2017-11-04 – szkielet gry

Poniższe opracowanie zawiera kompletny zakres materiału, który był realizowany na zajęciach Acodemia++ w dniu 4 listopada 2017. Materiał obejmuje zagadnienia tworzenia projektu, budowania scen oraz wstęp do tworzenia skryptów w środowisku Godot Engine. Stanowi podstawę do projektu, który w ramach zajęć będzie rozwijany o dodatkowe funkcjonalności Godot Engine. W załączonych materiałach znajduje się kompletna ścieżka postępowania przy budowie tego projektu (gry), a także gotowy projekt do uruchomienia. Zachęcamy do samodzielnej pracy. Ilość materiału wynika ze szczegółowego opisu czynności. Część z nich znana jest już z zajęć, część jest nowa. W ramach dalszych zajęć, opracowania będą bazowały na tych materiałach, a nasza praca i nauka będzie polegała na dodawaniu nowych rozwiązań oraz wykorzystywania już istniejących.

Szczecińskie Collegium Informatyczne
Szczecin – ul. Mazowiecka 13
www.sci.edu.pl

SZCZECIŃSKIE COLLEGIUM INFORMATYCZNE

www.acodemia.pl **e-mail:** info@acodemia.pl

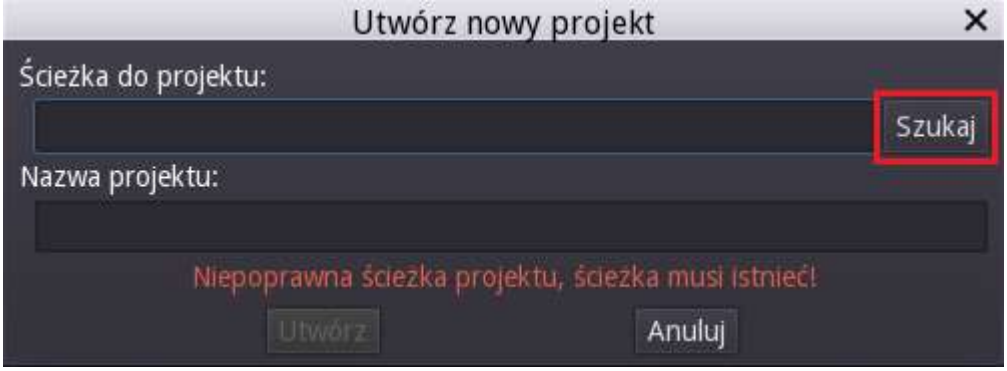
Tworzymy nowy projekt

Jeśli potrafisz już samodzielnie tworzyć i prawidłowo konfigurować projekt w środowisku **Godot Engine** – możesz pominąć poniższe kroki i przejść do kolejnego rozdziału. Jeśli nie, dla przypomnienia – poniżej przedstawiony jest cały proces prawidłowego tworzenia projektu. Dobrym rozwiązaniem jest utworzenie folderu, w którym będziemy trzymali nasze projektu zbudowane w środowisku **IDE** (ang. Integrated Development Environment) – **Godot Engine** ©.

Ilość danych przybywa. Pojawia się coraz więcej projektów, które tworzymy. Każdy programista musi zadbać o to, aby zapanować nad tymi wszystkimi danymi. Pamiętaj, aby każdy projekt miał swój osobny folder. Najwygodniej, jeśli nazwa tego folderu jest taka sama, jak nazwa projektu. Ważne jest również to, aby w tym folderze znajdowały się wszystkie dane, z których dany projekt korzysta.

1. Uruchom środowisko **Godot Engine**.
2. Wybierz opcję tworzenia nowego projektu – **Nowy projekt** 

.
3. Wybierz przycisk **Szukaj**, aby określić położenie projektu na dysku. Na zajęciach nasze projekty tworzymy na **Pulpicie**.



4. Wybierz przycisk **Utwórz katalog** Utwórz katalog.
5. Wpisz nazwę katalogu, w którym umieścisz swój projekt. W naszym przykładzie folder będzie miał nazwę **GAcodemia**. Zatwierdź przyciskiem **OK.**

6. Powrócisz do okna dialogowego.

7. W polu **Plik:** wpisz nazwę „**GAcodemia**”[1] i dopiero potem wciśnij przycisk **Otwórz**.

8. Ostatnie okno dialogowe to podsumowanie Twoich czynności. Wciśnij przycisk Utwórz **Utwórz**.

9. Umieść w folderze z projektem **GAcodemia** otrzymane pliki z grafiką, które pobrałeś ze strony.

[1] Nazwa folderu jest taka sama jak nazwa projektu, ale to dwa inne obiekty systemowe.

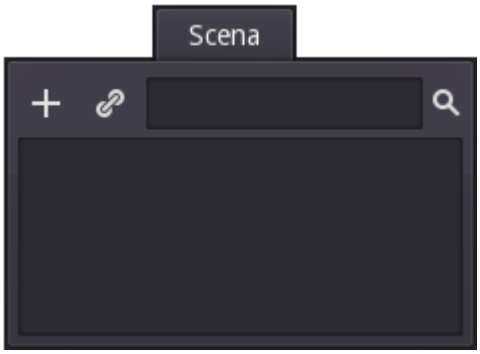
Szczecińskie Collegium Informatyczne
Szczecin – ul. Mazowiecka 13
www.sci.edu.pl

SZCZECIŃSKIE COLLEGIUM INFORMATYCZNE

www.acodemia.pl **e-mail:** info@acodemia.pl

Scena główna

Uruchom **Godot Engine**. Zaznacz projekt **GAcodemia** i wybierz opcję **Edycja**[2]. Odszukaj panel **Scena**. Przedstawia to poniższy rysunek.



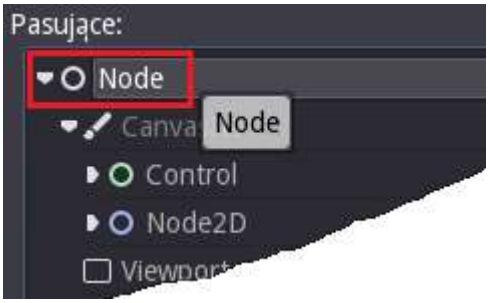
Jak widzisz, scena jest pusta. **Godot Engine** to środowisko, w którym pojęcie sceny jest bardzo istotne. Według dokumentacji – scena, to zbiór hierarchicznie zorganizowanych ze sobą węzłów, które tworzą drzewo zależności pomiędzy nimi. Na razie taka definicja może wydać Ci się niezrozumiała i skomplikowana, ale na kolejnych zajęciach zapoznasz się z tymi pojęciami. Pamiętaj, że tylko jedna scena w obrębie projektu może być sceną główną. Od tej sceny należy rozpocząć tworzenie projektu. Jakkolwiek, można potem zdecydować samemu, która scena będzie sceną główną. Musisz wiedzieć, że scena główna to ta scena, od której rozpoczyna się działanie tworzonego programu – gry – po jego uruchomieniu. Dodam, że całe sceny można kopiować do innych projektów, a nawet tworzyć całe biblioteki, aby wielokrotnie je wykorzystywać. Należy także pamiętać, że przenosząc sceny, uwzględnić należy także dane, o ile sceny z nich korzystają. Dane projektu w **Godot**

[2] Pomiń te kroki, jeśli masz już uruchomione środowisko Godot Engine.

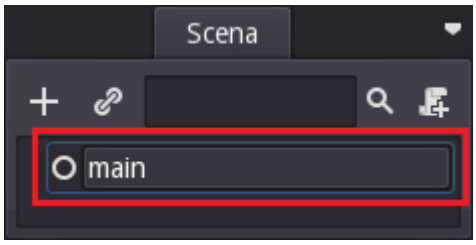
Engine można grupować w foldery, o czym już decyduje projektant. Ważne, aby wszystkie były w folderze projektu. W naszym przypadku folderze **GAcodemia**.

Dodamy teraz do sceny obiekt, który będzie reprezentował główny węzeł sceny, w tym przypadku główny węzeł sceny głównej. Wciśnij ikonę ze znakiem plus.

Uruchomi się okno dialogowe, za pomocą którego wybierzemy interesujący nas obiekt. Wybieramy obiekt **Node**, główny węzeł w hierarchii obiektów, który nosi nazwę korzenia głównego (ang. root).

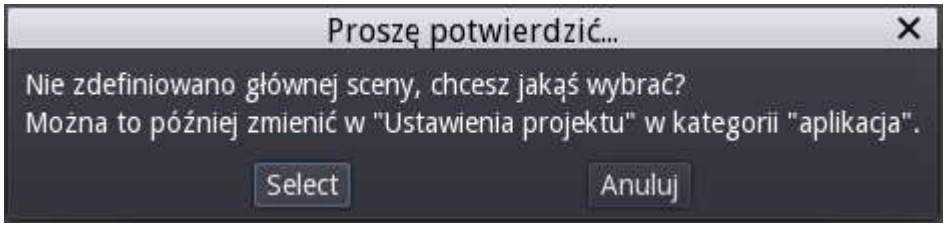


Wybór zatwierdzamy przyciskiem **Utwórz**. Po wykonaniu tych czynności na liście obiektów sceny pojawi się ikona **Node**. Zmień nazwę tego węzła na **main**.



Uruchom program. Wciśnij  symbol Play. Zostaniesz poinformowany, że scena nie została zapisana. Potwierdź komunikat, a scenie nadaj nazwę **main.tscn**.[3]

Ponownie uruchom grę. Zostaniesz poinformowany, że nie została wskazana scena główna.

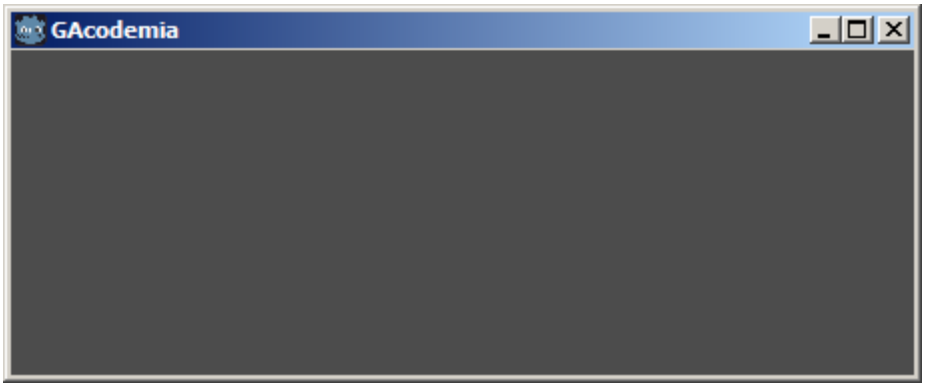


Wybierz **Select**, a następnie wybierz plik ze sceną **main.tscn**.

Ponownie uruchom program. Okno z działającym programem – grą powinno przypominać to na poniższym zrzucie ekranowym[4]. Aby powrócić do trybu

[3] Pliki nazw scen mają rozszerzenie **tscn**.

projektowania, zakończ działanie aplikacji, wciskając ikonę z krzyżykiem. Ważne jest także to, że środowisko **Godot Engine**, nie pozwoli zapisać sceny pustej, czyli takiej, która nie posiada dodanego obiektu – węzła głównego (ang. root).



Rysunek 1 Uruchomione okno gry

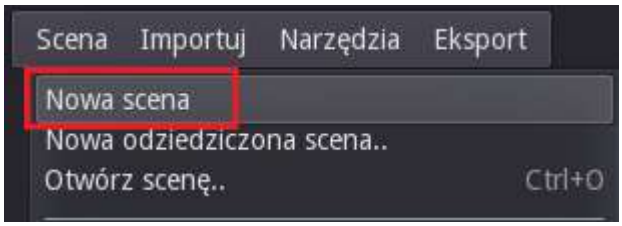
Posiadamy poprawnie skonfigurowany projekt, który prawidłowo się uruchamia. Mamy scenę główną, która zawiera węzeł główny. Zajmiemy się teraz budowaniem poszczególnych scen, których **instancje**, będą składowymi sceny głównej. **Instancja**[5] to egzemplarz sceny. Scenę budujemy tylko jeden raz, a potem możemy do woli z niej korzystać, tworząc tyle obiektów danego typu – sceny – ile przewiduje projekt. To znacznie uprasza i skraca czas tworzenia programów. Jest to bezpośrednie nawiązanie do paradygmatu programowania obiektowego. Instancja to schemat, przepis, wzorzec do powielania.

[4] Jeśli zostaniesz poinformowany, że nie wybrano dla projektu sceny głównej – wybierz scenę **main.tscn**.
[5] https://pl.wikipedia.org/wiki/Instancja_(programowanie)

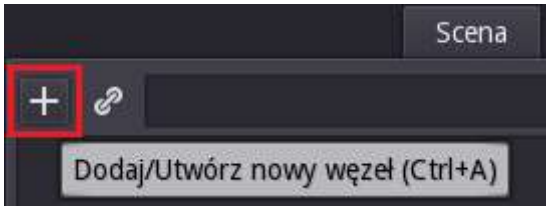
Tworzymy scenę Platform

Scena **Platform**, którą teraz zbudujemy będzie reprezentowała element platformy – ciała sztywnego, z zaimplementowaną fizyką statyczną. Rolą tego obiektu (tej sceny), jest funkcjonalność, za pomocą której będziemy budowali ścieżki, labirynty, przeszkody, inne elementy świata, po którym będzie poruszał się nasz **Gracz**,

1. Z menu **Scena**, wybieramy opcję, za pomocą której dodamy do projektu nową scenę.



2. W oknie reprezentującym scenę wybieramy ikonę  , która służy do dodawania obiektów dla aktualnie projektowanej sceny.



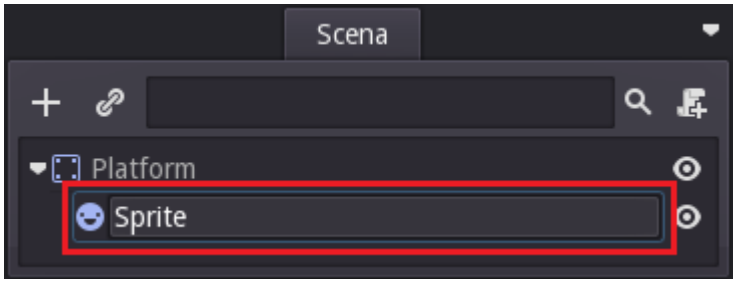
3. W oknie zawierającym hierarchicznie ułożone, dostępne w środowisku **Godot Engine**, węzły reprezentujące określone obiekty – wybieramy **StaticBody2D**. W oknie przedstawiającym składniki sceny widać dodany przez nas obiekt.

4. Zmieniamy jego nazwę na **Platform**.

5. Teraz musimy dla tego węzła dodać jego węzły potomne – dzieci (ang. **child**). W informatyce stosujemy pojęcie węzeł rodzicielski (nadrzędny) – (ang. **parent**), który może posiadać węzły podrzędne. Zaznaczamy zatem węzeł główny sceny – (ang. **root node**) 

.

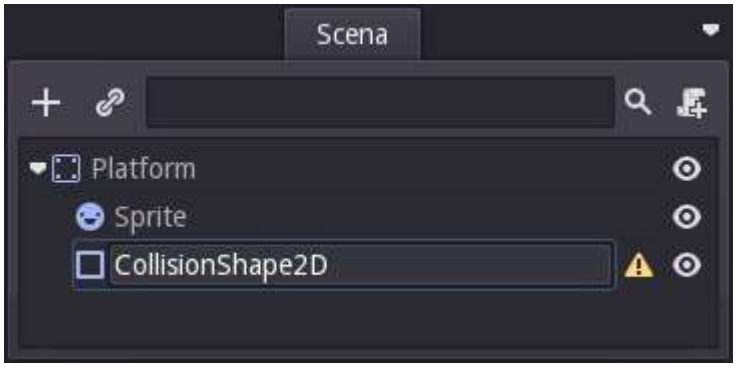
6. Dodamy teraz do niego dwa węzły podrzędne. Najpierw zadbamy o to, aby nasza przeszkoda miała reprezentację graficzną – dodamy obiekt **duszka** (ang. **Sprite**).
7. Ponownie wybieramy ikonę  , a z listy dostępnych obiektów wybieramy obiekt odpowiedzialny za wyświetlanie grafiki – **Sprite**. Nasze okno Scena zawiera teraz następujące węzły.



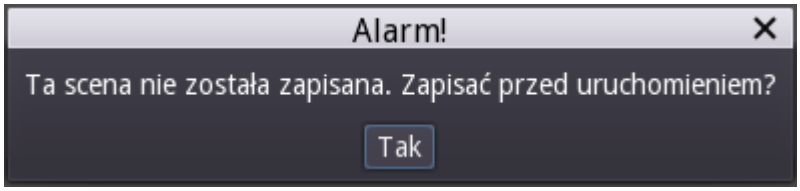
8. Ponownie wybieramy ikonę  , pamiętając, aby węzłem wybranym był węzeł 

i wybieramy obiekt, który będzie odpowiadał za obsługę kolizji. Grafika platformy jest obszarem prostokątnym, zatem wybieramy obiekt odpowiadający obszarowi prostokątnemu.

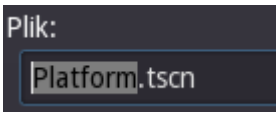
9. Z listy dostępnym obiektów wybieramy **CollisionShape2D**, co przedstawia poniższy zrzut ekranowy. Nasza scena zawiera już komplet dodanych węzłów.



Próba uruchomienia programu, spowoduje wyświetlenie komunikatu, ze nasza scena nie została zapisana!



Zatwierdzamy komunikat i zapisujemy ją pod nazwą **Platform.tscn**.



Zapisujemy 

.

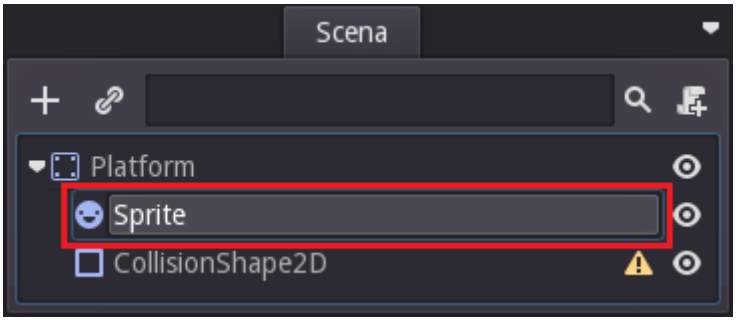
Szczecińskie Collegium Informatyczne
Szczecin – ul. Mazowiecka 13
www.sci.edu.pl

SZCZECIŃSKIE COLLEGIUM INFORMATYCZNE

www.acodemia.pl **e-mail:** info@acodemia.pl

Konfiguracja sceny Platform

1. Dla sceny **Platform.tscn** zaznaczamy węzeł, który odpowiada za reprezentację graficzną przeszkody – **Sprite**.



2. W oknie Inspektor, na którym wyświetlane są cechy obiektu, wybieramy Texture, rozwijamy listę <null>, a następnie wybieramy opcję ładowanie tekstury z pliku Load.

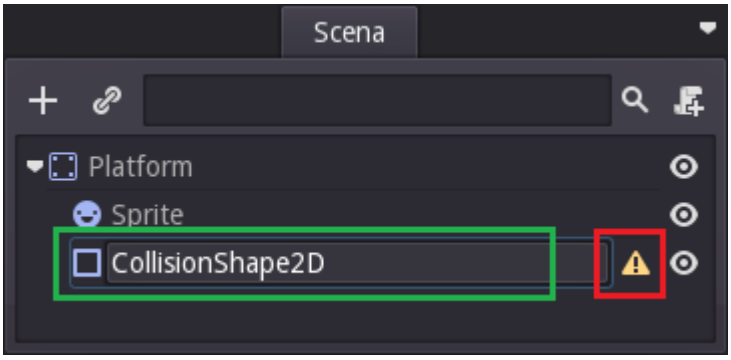
3. W oknie dialogowym będzie widoczna grafika, plik **Platform.png**.

4. Zaznaczamy ten plik i wybieramy przycisk **Otwórz**. Nasz duszek ma teraz założoną grafikę.

Teraz zajmiemy się węzłem **CollisionShape2D**, dla którego określimy prostokątny obszar otaczający naszego **Sprite**. Dzięki temu zaprojektujemy kolizję

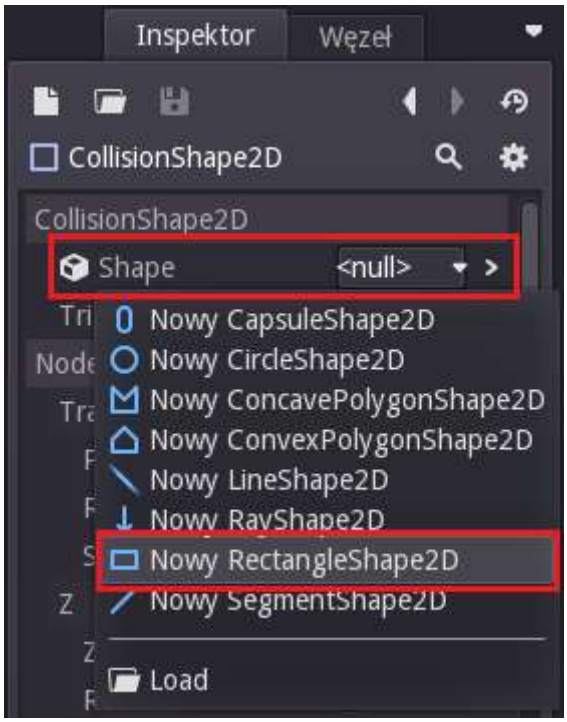
tego obiektu – ściślej rzecz ujmując – nasza scena **Platform** będzie reagowała na każde ciało, które go „dotknie". Lub jak kto woli – nastąpi reakcja ciała z przeszkodą.

Zaznaczamy węzeł **CollisionShape2D**. Po prawej stronie tego węzła, w oknie sceny można zauważyć ikonę ostrzeżenia ⚠. Informacja ta wskazuje, że musimy zdefiniować prostokątny obszar kolizji. W naszym przypadku, rozmiar tego prostokątnego obszaru będzie taki sam jak rozmiar **Sprite**, o rozmiarze którego decyduje z kolei rozmiar tekstury.

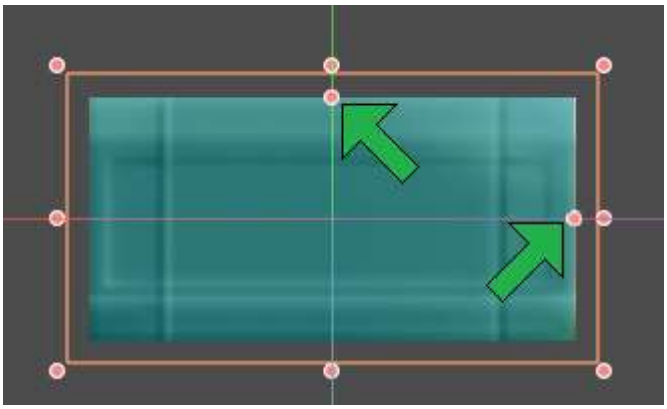


Zaznaczamy węzeł **CollisionShape2D**. Następnie w oknie **Inspektor**, dla opcji **Shape** (ang. Kształt), rozwijamy listę i wybieramy kształt prostokątny – **Nowy RectangleShape2D** 

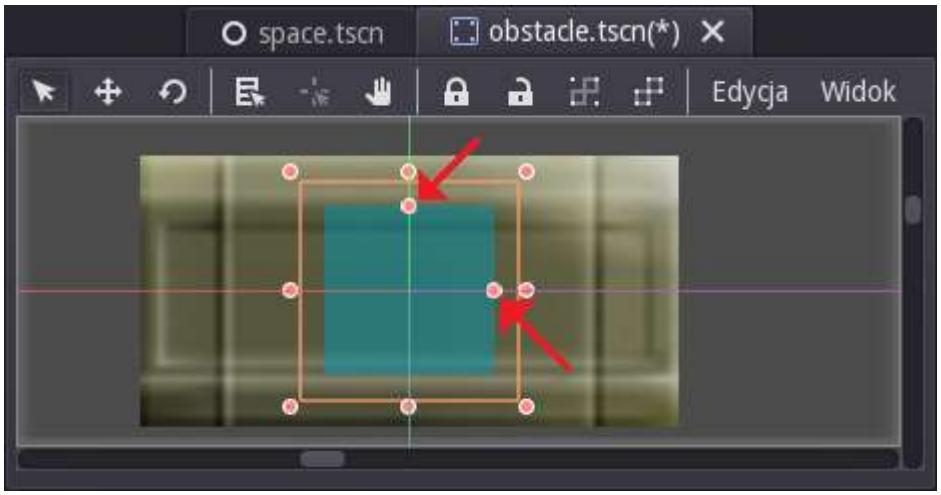
.
Przedstawia to poniższa ilustracja.



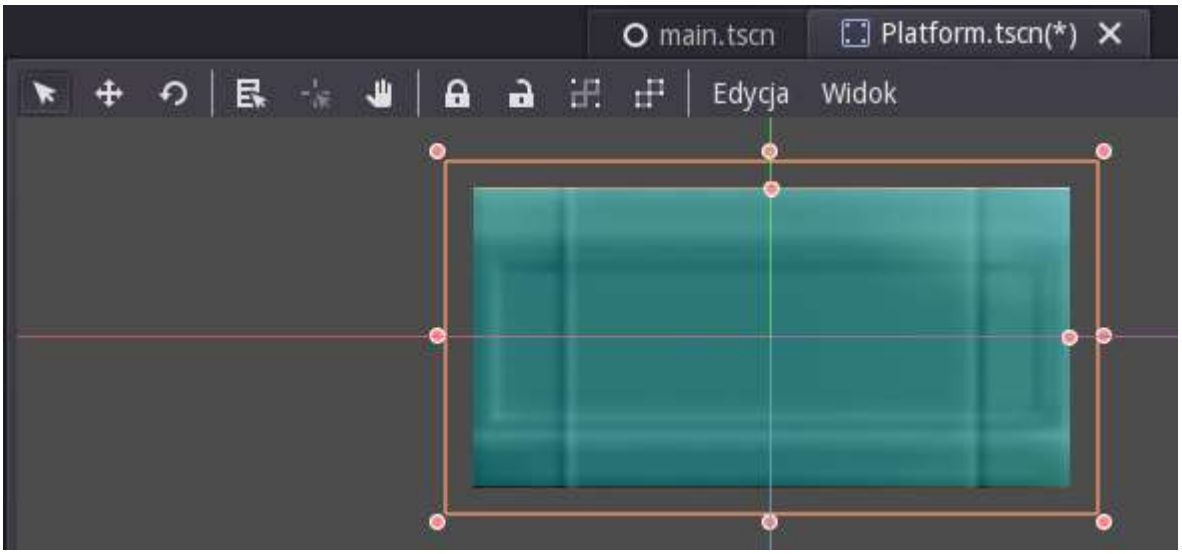
Po wykonaniu tego polecenia, zniknie ikona ostrzeżenia ⚠, a nasz obiekt na scenie będzie wyglądał tak jak na ilustracji poniżej. Niebieski, przezroczysty obszar to właśnie obszar kolizji. Pamiętaj, że możesz zmieniać skalę widoku projektowanej sceny, co jest bardzo wygodne, szczególnie, gdy chcemy coś precyzyjnie zaznaczyć lub zmodyfikować. Można to najwygodniej zrealizować rolką myszy, lub wybrać opcję **Widok** z panelu z narzędziami edycji sceny.



Zielona strzałka wskazuje na punkty nawigacyjne, za pomocą których zdefiniujemy obszar kolizji. Punkty te są aktywne, gdy w oknie sceny zaznaczymy obiekt **CollisionShape2D**.



Za pomocą punktów nawigacyjnych, które zostały zaznaczone **czerwonymi strzałkami**, modyfikujemy rozmiar prostokątnego obszaru kolizji – tak aby miał wielkość **Sprite**. Pamiętaj, że musimy wskazać węzeł **CollisionShape2D**, aby ta funkcjonalność była dostępna i aktywna. Modyfikujemy teraz rozmiar, aby uzyskać taki efekt, jak na poniższej ilustracji.



Rysunek 2 Poprawnie zdefiniowany prostokątny obszar kolizji

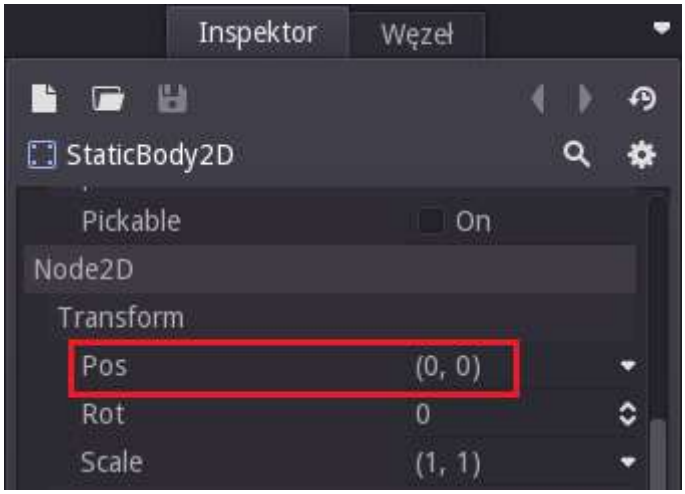
Testujemy scenę Platform

Uruchamiamy scenę. Pamiętaj, że ikona służy do uruchamia sceny – klawisz **F6**, którą modyfikujemy. Natomiast ikona , to ikona uruchamiająca grę – scenę główną – klawisz **F5**. Warto zapamiętać te skróty klawiszowe, tym bardziej, że są one wykorzystywane w innych środowiskach programistycznych i przypisane są im podobne funkcjonalności.

Po uruchomieniu sceny okno gry(sceny) wygląda jak poniżej[6].

[6] Niektóre zrzuty ekranowe widoczne w tym dokumencie są zmodyfikowane, aby nie były zbyt obszerne. Na Twoim komputerze mogą mieć inny wygląd.

Zwróć uwagę, że widzimy tylko jedną ćwiartkę obiektu sceny. Dzieje się tak dlatego, że projektowana scena posiada swój lokalny układ współrzędnych, a nasz obiekt przeszkoda, dokładnie jego środek geometryczny – znajduje się w początku tego układu współrzędnych.
Pozycja: (X = 0, Y = 0), co można zobaczyć w oknie **Inspektor**.

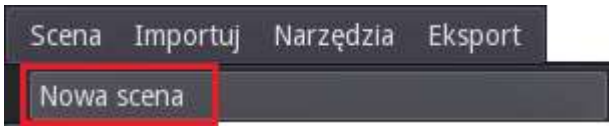


Jeśli poprawnie wszystko zostało zrealizowane – oznacza to, że scena, która swoją funkcjonalnością odpowiada platformie – jest ukończona. Zapisz scenę (sceny), wybierając odpowiednią opcję z menu. Pamiętaj, że jeśli projektujesz scenę, to dla wygody, nie zmieniaj pozycji jej składowych (węzłów), względem układu współrzędnych. Niech ich pozycja będzie środkiem układu współrzędnych (0, 0).

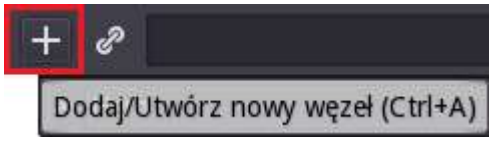
Scena Player

Zbudujemy teraz scenę, która będzie reprezentować gracza. Będzie to obiekt posiadający animowaną grafikę, zaimplementowaną fizykę, kolizje oraz docelowo skrypt, który będzie zmieniał zachowanie obiektu w wyniku kolizji tego obiektu z innymi obiektami sceny (z platformą). Poniżej wymieniono czynności, które są niezbędne do prawidłowego działania sceny **Player**.

1. Z menu wybieramy opcję **Scena -> Nowa scena**

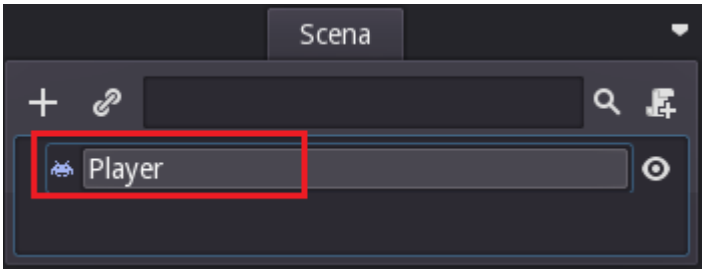


2. Do nowej sceny dodajemy jej węzeł główny. Wybieramy ikonę z plusem.



3. Następnie szukamy węzła reprezentującego obiekt **KinematicBody2D**.

4. W panelu **Scena**, zmieniamy nazwę tego węzła na **Player**.



Animacja poklatkowa

Dodamy teraz do naszej sceny obiekt odpowiadający za grafikę. Tym razem wykorzystamy obiekt **AnimatedSprite**. Zanim to zrobimy, parę zdań o animacji poklatkowej.

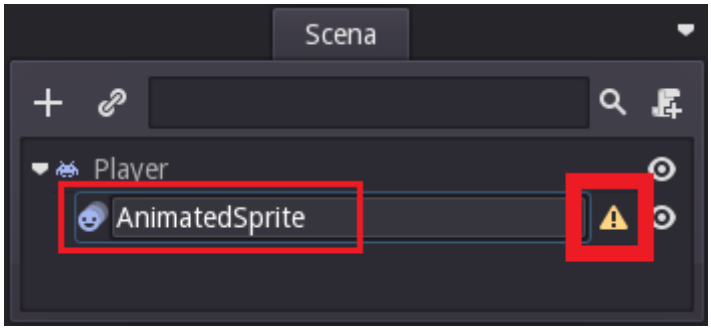
Definicja: **Animacja poklatkowa** (ang. Stop motion, ang. Cutout animation) – animacja stworzona na podstawie zarejestrowanych przez kamerę osobnych klatek, które mogą być rysunkami, zdjęciami, lub innymi obiektami rejestrowanymi przez kamerę. Odtworzona, jako sekwencja (film) klatka po klatce, oddaje wrażenie efektu ożywienia – ruchu, zapisu pewnego wycinka otaczającej nas rzeczywistości.

Technika animacji ma bardzo wiele lat. Od dziesięcioleci jest udoskonalana. Wykorzystywana w filmach rysunkowych, popularnych kreskówkach. Początkowo każda klatka rysowana była metodami tradycyjnymi, czyli na papierze, a następnie fotografowana jako osobna klatka na kliszy filmowej. Dzisiaj całą pracę wykonuje się na komputerze. Tę właśnie technikę, której podstawowa zasada znana jest od wielu lat, wykorzystamy na potrzeby naszej gry.

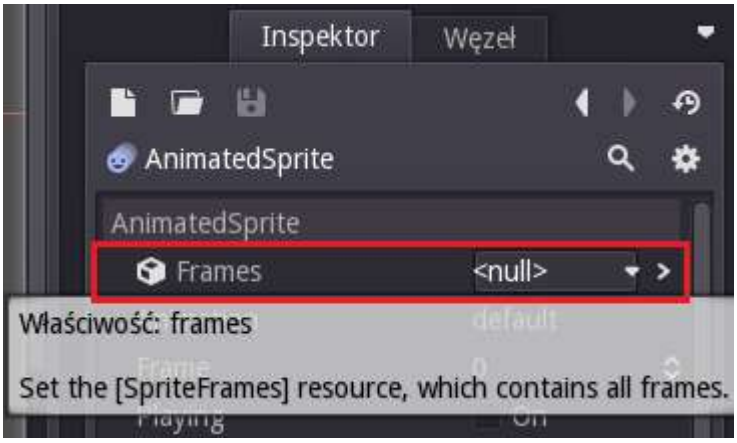
Na powyższym zrzucie ekranowym widzimy sekwencję ośmiu klatek animacji. Przyglądając się uważnie poszczególnym klatkom, licząc od lewej strony, widzimy, że nasz insekt (**Player**) porusza odnóżami. Różnice poszczególnych klatek, które zostaną zauważone, gdy wyświetlimy nasz obiekt na scenie. Każda klatka animacji, – jako osobny plik graficzny, została umieszczona w folderze projektu.

5. Dodaj do sceny Player węzeł **AnimatedSprite** 

.
Nasza scena zawiera dodany węzeł. Widzimy ikonę z ostrzeżeniem – wskazówką, że musimy zdefiniować klatki animacji.



6. W **Inspektorze**, dla wskazanego obiektu **AnimatedSprite**, wybieramy określenie **Frames**, jak zostało to przedstawione na poniższym zrzucie ekranowym.

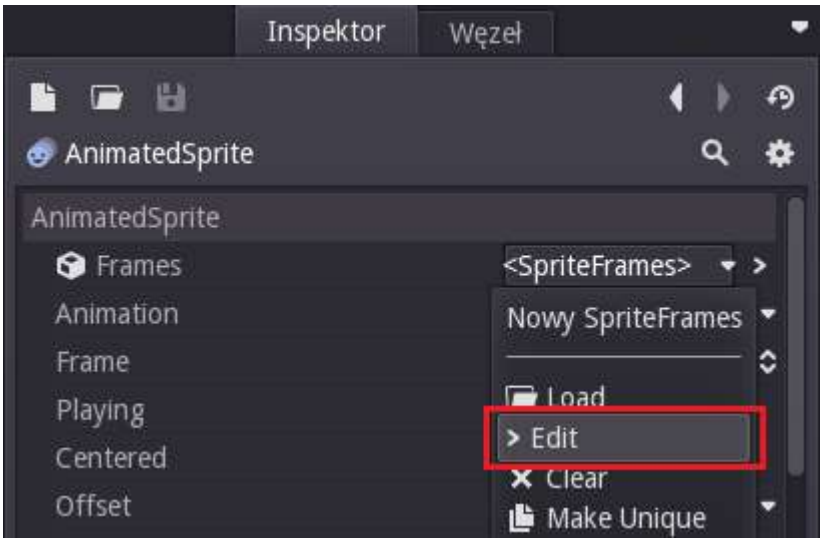


7. Wybieramy opcję, **Nowy SpriteFrames**, za pomocą której zdefiniujemy zbiór klatek animacji.

Rysunek 3 Wybieramy nowy zestaw klatek animacji

8. Następie rozwijamy listę **SpriteFrames**.

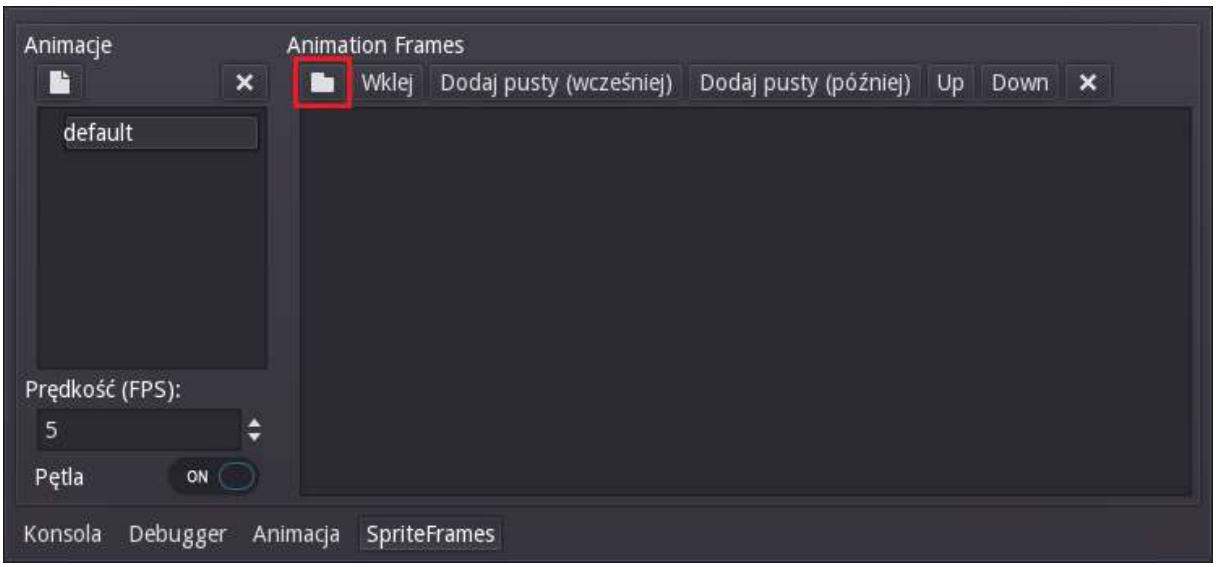
9. Wybieramy opcję **Edit**.



Po wybraniu opcji **Edit** uruchomi się panel, w którym zdefiniujemy zasób, – czyli zbiór klatek animacji. Przedstawia to poniższy zrzut ekranowy.

Rysunek 4 Panel zarządzania animacją poklatkową

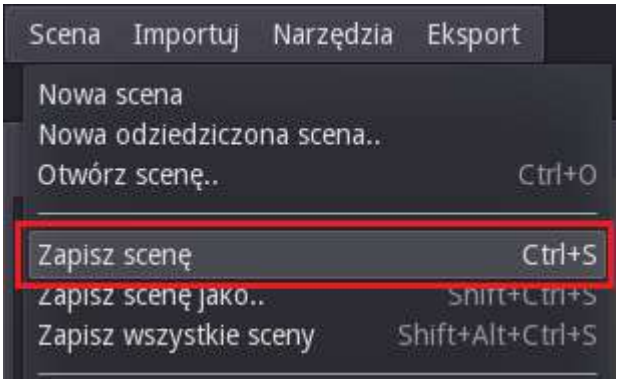
10. Za pomocą opcji ładowania zasobów – wczytamy klatki animacji, które znajdują się jako osobne pliki z folderze projektu. W tym celu wybieramy ikonę odpowiadającą za wczytanie zasobów.



11. Teraz zaznaczamy osiem klatek animacji. Aby od razu zaznaczyć je wszystkie – zaznaczenie musimy zrobić mając wciśnięty klawisz **Shift**.

W naszym przykładzie poszczególne klatki – ich nazwy zawierają numerację. Dodatkowo wszystkie są tej samej wielkości. Wiedz, że możesz w dowolny sposób tworzyć poszczególne klatki. Dodawać nowe. Wstawiać na koniec, przesuwać – modyfikować je dowolnie.

12. Zatwierdzamy przyciskiem **Otwórz**. Zanim przetestujemy scenę musimy ją zapisać. Wybieramy z menu **Scena – Zapisz scenę**.



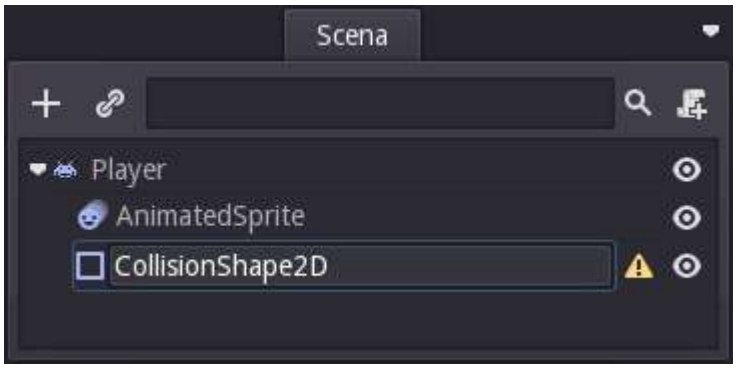
13. Zapisujemy scenę pod nazwą **Player.tscn**. Najpierw włączymy odtwarzanie animacji. W tym celu, w **Inspektorze** wyszukujemy opcję **Playing**.

14. Uaktywniamy ją.

Widzimy, że jeśli opcja odtwarzania jest aktywna – licznik klatek **Frame** – pokazuje aktualnie odtwarzaną klatkę. Zwróć uwagę, że odtwarzane klatki animacji są ponumerowane od **0** do **7**. Nasza animacja składa się z ośmiu klatek. W informatyce, szczególnie w kontekście języków programowania, zbiór elementów danego typu nosi nazwę tablicy. O tablicach dowiesz się więcej podczas zajęć programowania w języku **C/C++**. Na razie musimy wiedzieć, że elementy tablicy posiadają przypisane im indeksy, a te numerujemy od zera. W naszym przypadku elementami tablicy są poszczególne klatki animacji (grafika).

Można zauważyć, że animacja się odtwarza. W dodatku widać efekt działania sceny, bez konieczności uruchamia jej. Środowisko Godot Engine posiada mechanizm działania pętli czasu rzeczywistego w trybie edycji danych. O pętli czasu rzeczywistego także dowiesz się podczas nauki **C/C++** – napiszesz ją samodzielnie.

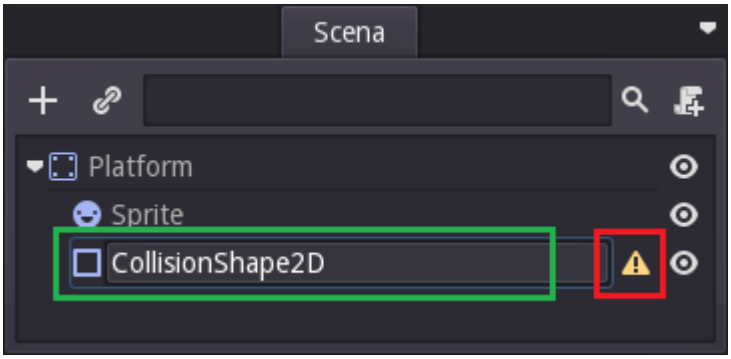
15. Dodajemy do sceny węzeł obsługujący kolizje. Z listy dostępnym obiektów wybieramy **CollisionShape2D**, co przedstawia poniższy zrzut ekranowy. Nasza scena zawiera już komplet dodanych węzłów.



Konfiguracja sceny Player

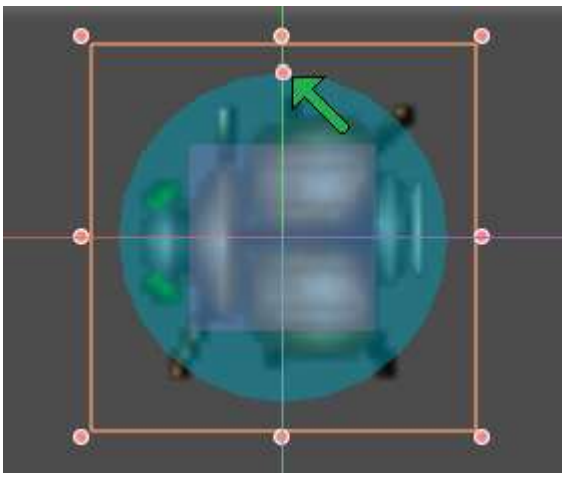
Podobnie jak w przypadku sceny **Platform** – zajmiemy się konfiguracją węzła odpowiadającego za kolizję – **CollisionShape2D**. Tym razem zdefiniujemy okrągły obszar kolizji ponieważ **Sprite** sceny **Player**, najbardziej temu kształtowi odpowiada.

1. Zaznaczamy węzeł **CollisionShape2D**



2. W oknie **Inspektor**, dla opcji **Shape** (ang. Kształt), rozwijamy listę i wybieramy kształt okrągły – **Nowy CircleShape2D** 
.

3. Po wykonaniu tego polecenia, zniknie ikona ostrzeżenia , a nasz obiekt na scenie będzie wyglądał tak jak na ilustracji poniżej. Niebieski, przezroczysty obszar to właśnie obszar kolizji.

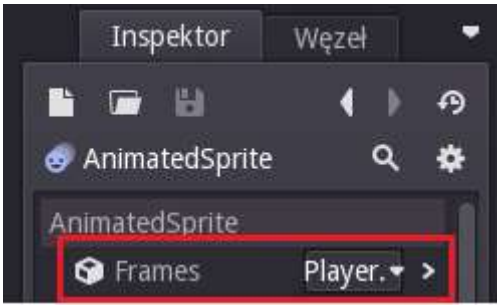


4. Zielona strzałka wskazuje na punkt nawigacyjny, za pomocą którego zdefiniujemy obszar kolizji. Punkt ten są aktywny, gdy w oknie sceny zaznaczymy obiekt **CollisionShape2D**. Modyfikujemy rozmiar okrągłego obszaru kolizji – tak aby miał wielkość **Sprite**.

Testujemy scenę Player

Uruchamiamy scenę  . Jeśli nie została wcześniej zapisana, zapisujemy ją pod nazwą **Player.tscn**. Po uruchomieniu sceny okno gry(sceny) wygląda jak poniżej. Można zauważyć, że animacja poklatkowa jest odtwarzana.

Zaznaczając dla sceny Player węzeł **AnimatedSprite**, a następnie w Inspektorze rozwijając listę dla opcji **Frames** i wybierając **Edit** – przejdziemy do panelu zarządzania animacją, gdzie między innymi można ustawić prędkość odtwarzania klatek animacji. Ustalić, czy animacja ma się zapętlać (czyli po odtworzeniu ostatniej klatki, ponownie odtwarzana jest klatka pierwsza. Zarządzać innymi funkcjonalnościami.



Szczecińskie Collegium Informatyczne
Szczecin – ul. Mazowiecka 13
www.sci.edu.pl

SZCZECIŃSKIE COLLEGIUM INFORMATYCZNE

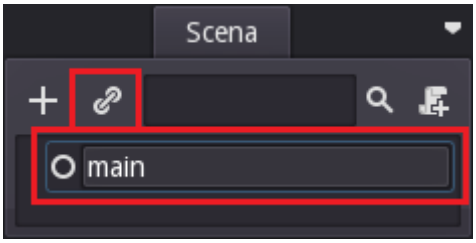
www.acodemia.pl **e-mail:** info@acodemia.pl

Budujemy grę – scena główna

Mamy gotowe dwie sceny: **Platform** oraz **Player**. Dodamy teraz do sceny głównej jedną instancję sceny oraz kilka instancji sceny **Platform**.

1. Wybieramy scenę główną **main.tscn** 

. Zaznaczamy węzeł główny (**main**) i za pomocą ikony  , za pomocą której dodajemy do sceny instancję sceny **Player**.



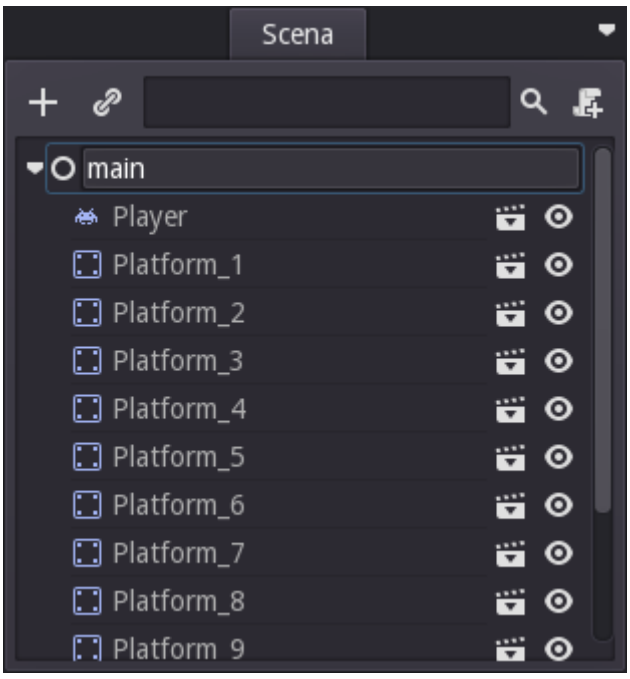
2. W oknie dialogowym wybieramy scenę **Player.tscn** i potwierdzamy przyciskiem **Otwórz**.

3. Podobnie postępujemy, aby dodać do sceny instancje sceny **Platform**.

Zaprojektuj scenę według swojego pomysłu. Pamiętaj, że warto posługiwać się mechanizmem duplikowania, co znaczenie ułatwia i przyspiesza projektowanie scen. Aby utworzyć duplikat, zaznaczamy na panelu Scena interesujący nas węzeł, Wciskamy prawy przycisk myszy i wybieramy opcję **Duplikuj** 

, lub korzystamy ze skrótu **Ctrl+D**.

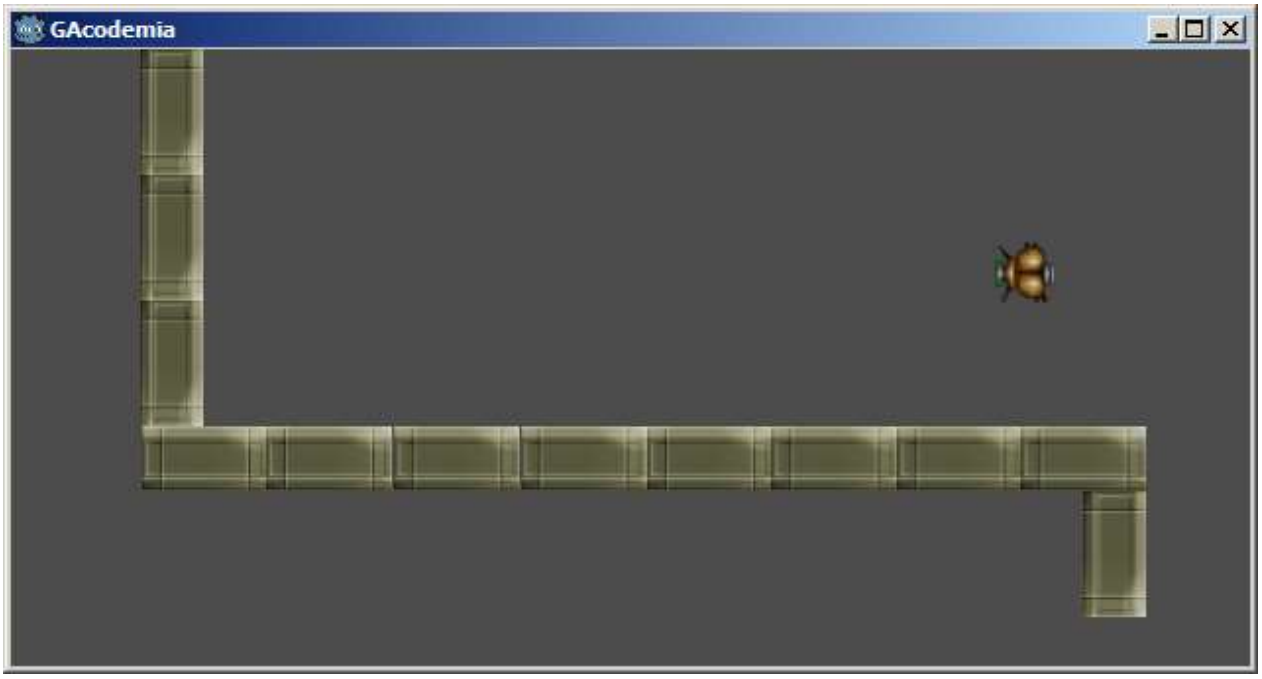
W oknie sceny zostanie zarejestrowany duplikat. **Godot Engine** automatycznie będzie nadawał nazwy obiektom w hierarchii obiektów danej sceny, dopisując do nazwy kolejne liczby całkowite.



Rysunek 5 Zduplikowane obiekty sceny **Platform**

Aby zobaczyć dodany obiekt (duplikat) na scenie, musimy go przesunąć, ponieważ po utworzeniu duplikatu, ma on takie samo położenie jak obiekt, na podstawie którego został utworzony.

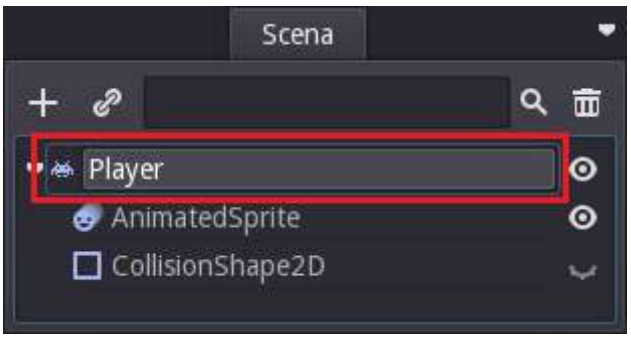
Poniżej przykładowy wygląd sceny, którą zaprojektowałem, w oknie gry po jej uruchomieniu.



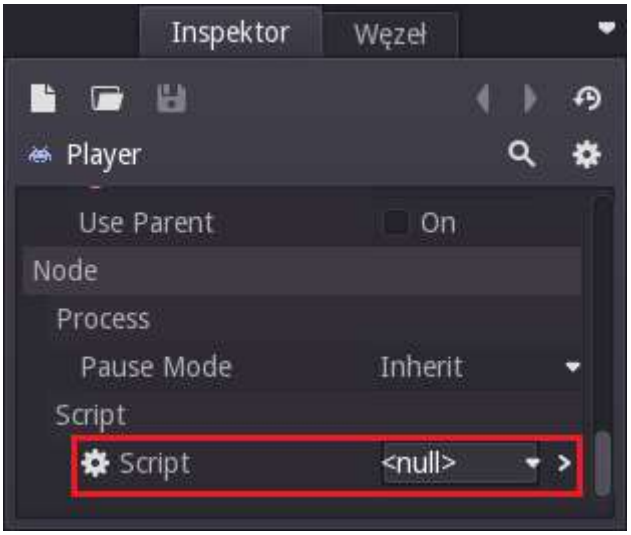
Nasz pierwszy skrypt

W folderze z projektem znajduje się plik Player.gd. Godot Engine posiada wbudowany język skryptowy. O skryptach powiemy sobie więcej na kolejnych zajęciach projektu Acodemia++. Będziemy je sami tworzyli, modyfikowali istniejące. Podłączymy skrypt do sceny **Player**.

1. Otwórz scenę Player – plik **Player.tscn**.
2. Zaznacz w panelu Scena jej węzeł główny – **Player**



3. W **Inspektorze** przejdź do opcji **Script**



4. Rozwiń listę i wybierz **Load.**

5. Wybierz plik skryptu **Player.gd** i otwórz go.

6. Uruchom grę i sprawdź, czy nasz Gracz (**Player**) porusza się, gdy wciskamy przyciski strzałek. Jeśli tak, oznacza to, że prawidłowo skojarzyliśmy plik ze skryptem ze sceną.
7. Sprawdź, że jeśli nasz **Player** (insekt…) najedzie na element sceny – **Platform**, animacja się zatrzymuje. Za to wszystko właśnie odpowiada skrypt.

   Poniżej treść skryptu. Na kolejnych zajęciach będziemy wielokrotnie korzystali ze skryptów. Modyfikowali je. Pisali własne. W ten sposób będziemy poznawali język programowania.

```
extends KinematicBody2D

const MOTION_SPEED = 160 # Pixels/seconds

func _fixed_process(delta):

	var motion = Vector2()

	if (Input.is_action_pressed("ui_up")):
		motion += Vector2(0, -1)
	if (Input.is_action_pressed("ui_down")):
		motion += Vector2(0, 1)
	if (Input.is_action_pressed("ui_left")):
		motion += Vector2(-1, 0)
	if (Input.is_action_pressed("ui_right")):
		motion += Vector2(1, 0)

	motion = motion.normalized() * MOTION_SPEED * delta
	motion = move(motion)

	if(is_colliding()):
		set_rotd(0)
		get_node("AnimatedSprite").stop()
		pass
	else:
		set_rotd(90)
		get_node("AnimatedSprite").play("worm_default_animation")
		pass

func _ready():
	set_fixed_process(true)
	pass

func _on_VisibilityNotifier2D_exit_screen():
	pass
```