POLITECHNIKA KOSZALIŃSKA
WYDZIAŁ TECHNOLOGII I EDUKACJI

Praca inżynierska
Mechatronika-Wzornictwo
Film, Gry i Animacje Cyfrowe

Edukacyjna gra ilustrująca działanie mechanicznych układów maszyn prostych

Educational game illustrating the operation of mechanical systems in simple machines

Jeremi Piotr Biernacki
Nr albumu: U-4420

Promotor: doktor inżynier
Andrzej Błażejewski

KOSZALIN 2015

STRESZCZENIE

Celem niniejszej pracy jest zaprojektowanie dwuwymiarowej gry edukacyjnej z zakresu fizyki dla szkoły ponadgimnazjalnej. Początkiem działania był przegląd i ocena narzędzi do tworzenia gier komputerowych. W pracy zaprezentowano niektóre cechy komputerowych gier edukacyjnych jako nowoczesnej i efektywnej metody nauczania. Na podstawie wcześniej wspominanej oceny narzędzi do tworzenia gier komputerowych wybrano silnik Godot Engine i omówiono jego możliwości. Opisano zaimplementowane w grze modele matematyczne. Kolejna część pracy została poświęcona procesowi tworzenia gry. W celu zrealizowania edukacyjnego aspektu gry wyposażono ją w trzy tryby. Ważnym dla pracy było wykorzystanie systemu kolizji i tzw. silnika fizyki.

Słowa kluczowe: programowanie gier komputerowych, silnik gry, grafika dwuwymiarowa, edukacja, fizyka, maszyny proste, Python, Godot Engine, Open Source.

SUMMARY

The aim of this study is to design a two-dimensional educational game for physics for secondary school. Begin of action was a review and assessment of tools for creating computer games. The study demonstrated some features of computer educational games as a modern and effective teaching methods. Based on the earlier mentioned evaluation of tools for creating computer games, the engine Godot Engine was selected and its features were discussed. Described implemented in the game mathematical models. Another part of the work has been devoted to the process of creating the game. In order to achieve the educational aspect, three modes were implemented in the game. Important feature of this work was the use of the so-called physics engine and system of collisions.

Keywords: programming computer games, game engine, two-dimensional graphics, education, physics, simple machines, Python, Godot Engine, Open Source.

Spis treści

1 Wprowadzenie

Wielu socjologów badających rozwój człowieka zgadza się, iż gry stanowią nieodłączny element kultury. Gry wręcz stanowią koło napędowe rozwoju cywilizacyjnego łącząc przekaz edukacyjny z kreatywnością i zabawą. Człowiek jest jedyną istotą, która gra dla przyjemności[1].

Według J. Huizinga, historyka i badacza gier: *„Gra to dobrowolna aktywność, która jest przez nas świadomie oddzielana od „zwyczajnego" świata jako „mniej poważna" i jednocześnie absorbująca gracza w sposób intensywny i całkowity. Czynność ta nie jest powiązana z korzyścią materialną. Gra odbywa się w swojej własnej przestrzenni, miejscu i czasie, zgodnie z ustalonymi regułami i przebiega w określonym porządku*[1]."

Współcześnie, oprócz kanonu gier i zabaw przekazywanych z pokolenia na pokolenie mamy kolejny wymiar gier, ściśle związanych z wymiarem technologicznym – są to gry komputerowe. Choć rozwijają się tak dynamicznie i stanowią istotny procent obrotu całej branży, to ich wykorzystanie w edukacji formalnej, szczególnie w grupie młodzieży gimnazjalnej i ponadgimnazjalnej jest znikome. Powszechnie stosowaną praktyką jest dołączanie materiałów multimedialnych i ćwiczeń na nośnikach CD do podręczników przedmiotowych. Można stwierdzić, że ten zabieg nie wykorzystuje w pełni potencjału tej technologii traktując gracza pasywnie. Przegląd dostępnych na rynku gier edukacyjnych to potwierdza. Dostępne gry to najczęściej multimedialne encyklopedie lub zbiory interaktywnych ćwiczeń np.: seria *Edurom* firmy *Young Digital Planet* czy serie *Fizyka* i *Matematyka* firmy *Didaktyka*.

1.1 Definicja silnika gry na podstawie Wikipedii

Polska literatura drukowana niestety nie nadąża za technologią informatyczną. W związku z czym autor jest zmuszony do korzystania ze źródeł internetowych.

Silnik gry zajmuje się interakcją elementów gry. Może posiadać wbudowane moduły grafiki, wejścia, sieci czy też sztucznej inteligencji i wykrywania kolizji między obiektami gry itd.
Może też korzystać z oddzielnych silników implementujących obsługę wymienionych modułów. [6].

1.2 Przegląd oprogramowania do tworzenia gier komputerowych

Dzisiejszy rynek programistyczny silników gier liczy sobie tysiące alternatywnych rozwiązań począwszy od tych najbardziej znanych jak id Tech bądź Unreal Engine, a kończąc na jeszcze niedocenianych produkcjach zdecydowanie mniejszych korporacji lub nawet niezależnych grup. Jednym z podstawowych kryteriów podziału tak licznej grupy oprogramowania jest ich

dostępność dla deweloperów*. Większość silników umożliwia wypróbowanie ich możliwości dzięki umieszczaniu w sieci kodu źródłowego silników na darmowej licencji (ang. *open source*). Opozycją do takiej postawy są programiści, których autorskie silniki gier objęte są ścisłą klauzulą poufności lub płatne. Przykładem są znane większości graczy bestsellerowe gry takie jak: God of War III, Metal Gear Solid 4 czy polski Wiedźmin (2 i 3 część).

Mając na uwadze powyżej wspominanie kryterium dostępności publicznej oraz możliwości zrealizowania założonych w tej pracy celów, zostanie zaprezentowany poniżej przegląd wybranego oprogramowania do tworzenia gier komputerowych:

- **Unity 3D** – jedno z najpopularniejszych narzędzi do tworzenia gier komputerowych 3D i 2D, oparte o technologię Mono – otwarty klon technologi .NET Microsoftu opracowany przez firmę Novell, dostępny na Windows i Mac OS X oraz za niedługo na Linuksie. Można kompilować gry pod te systemy oraz pod Systemy Mobline (Android i iOS) i konsole do gier (funkcjonalność płatna). Podstawowa funkcjonalność jest darmowa. Jest także możliwość wyboru języka, w którym można programować np.: C#, UnityScript (podobny do JavaScript) lub Java[7];
- **Unreal Engine 4** – narzędzie stworzone przez Epic Games. Od niedawna można korzystać z niego nieodpłatnie, a dodatkowo po zarejestrowaniu się na stronie firmy można zmieniać kod źródłowy. Twórca po komercyjnym wydaniu gry i przekroczeniu zysków wysokości $3000USD musi płacić 5% zysków co kwartał firmie Epic Games. Ten silnik jest głównie przeznaczony do tworzenia gier 3D. Obsługuje dużą liczbę platform: Microsoft Windows, Linux, Mac OS X, Xbox One, PlayStation 4, HTML5, iOS, Android. Można w nim programować w C++ lub korzystać z programowania wizualnego za pomocą tzw. Blueprint. Największym minusem tego silnika są jego spore wymagania sprzętowe. Zalecana przez jego twórców konfiguracja sprzętowa to komputer PC (Windows 7 64-bit) lub Mac (Mac OS X 10.9.2 lub nowszy) typu desktop, posiadający czterordzeniowy procesor o mocy obliczeniowej 2,5 GHz oraz kartę graficzna NVIDIA GeForce 470 GTX lub AMD Radeon 6870 HD oraz 8 gigabajtów pamięci operacyjnej[8];
- **Godot Game Engine** – od roku dostępny dla każdego jako oprogramowanie Open Source. Służy do tworzenia gier 2D i 3D. Obsługiwane platformy to: Windows, Mac OS X, Linux, HTML5, Android. Dostępny język to GDScript[9].

* Deweloper – często tak właśnie określa się programistów i pojęcia te będą używane zamiennie w tej pracy.

Analiza możliwości dostępnych silników zdecydowała o wyborze oprogramowania Godot Game Engine do realizacji dyplomowego projektu. Największą zaletą tego rozwiązania jest stopień zintegrowania ze sobą poszczególnych narzędzi. Daje to możliwości zaprogramowania przebiegu wydarzenia poprzez animacje i dostęp do zmiennych w skrypcie z poziomu graficznego edytora scen.

1.3 Stan gier komputerowych stosowanych w edukacji

Technologie ICT (ang. I*nformation and Communication Technologies*), zmieniają nasze codzienne życie w każdym niemal wymiarze, szczególnie zagarniając sferę rozrywki i komunikacji. Stosunkowo w niewielkim stopniu są wykorzystywane w edukacji na poziomie podstawowym, gimnazjalnym i ponadgimnazjalnym. Tymczasem proces edukacji, by był efektywny, wymaga obecnie stosowania nowych metod, form i środków dydaktycznych. Aktualnie w szkołach komputery i programy używane są do nauczania informatyki. Inicjatywa edukacyjna G4LI (ang. *Games for Learning Institute*) wskazuje na wyraźne większe zaangażowanie młodych ludzi w uczenie się, gdy mogą korzystać z interaktywnych form, szczególnie z użyciem komputera. Nad tą ideą współpracują głównie Uniwersytet Karoliny Północnej i Carnegie Mellon University. Ten projekt badawczy pokazuje, że gry:

- uczą rozwiązywania skomplikowanych problemów,
- wspierają nauczanie indywidualne,
- są pomostem pomiędzy nauką w szkole i w domu,
- pozwalają łatwo śledzić postępy i wyniki w nauce[10].

Użycie komputerowych programów, które zwykle kojarzą się z wymiarem rozrywkowym powodują większą motywację i zaangażowanie.

Tymczasem stosowane w szkole metody asymilacji wiedzy, czyli wykład, dyskusja i praca z książką, koncentrują się głównie na przekazie słownym, angażując tylko zmysł słuchu o niewielkiej sile oddziaływania.

„Ta grupa metod, zwanych też metodami podającymi, znajduje rozległe zastosowanie w szkole. Cała praca sprowadza się do doboru treści oraz sposobu jej przekazywania przy czym od charakteru treści i metody jej „podania” zależy asymilacja wiedzy przez ucznia i trwałość jej zapamiętania oraz stopień zmotywowania uczniów i stopnia upoglądowienia nauczania.

Do metod zalicza się: pogadankę, dyskusję, wykład oraz pracę z książką[2].”

Poparcie wykładu obrazem angażuje kolejny zmysł - wzrok, zwiększając siłę zapamiętywania. Obraz może być wprowadzony za pomocą komputera. Najefektywniejsza nauka zachodzi jednak wtedy, gdy uczeń ma okazję wykazać się aktywnością w obszarze danego zagadnienia, czyli dokonać transferu zdobytej wiedzy i praktycznie ją wykorzystać. Interaktywne programy angażując większą ilość zmysłów stają się istotnym wsparciem procesu dydaktycznego i jako takie zasługują na uwagę. Szczególnie istotne jest urozmaicenie zajęć z tych przedmiotów, które od lat są wyzwaniem dla szkół. O trudności w nauczaniu takich przedmiotów jak matematyka i fizyka świadczy niski poziom punktacji i zdawalności na egzaminach gimnazjalnych i maturalnych. Dane z roku 2015 ilustrują tę tendencję: jak podaje portal gazety Newsweek (newsweek.pl) egzamin gimnazjalny w zakresie matematyki uczniowie pisali średnio na 48% punktów, a z fizyki otrzymywali średnio 50% punktów[11]; natomiast egzamin maturalny według danych portalu gazety Fakt (fakt.pl): matematykę zdało 82% uczniów a ci, którzy przystąpili do zdawania fizyki uzyskali wynik 44% punktów[12]). Każdego roku po ogłoszeniu wyników egzaminów przechodzi przez media fala krytycznych komentarzy odnośnie tego zjawiska jak np. opinia z portalu dla maturzystów perspektywy.pl:

„*A budzący najwięcej emocji – … egzamin z matematyki wypadł dużo słabiej niż rok temu – nie zaliczyło 21 proc. osób, w 2010 r. ten odsetek był mniejszy o 8 %*[13].”

Na rynku dostępnych jest wiele gier edukacyjnych. Jednak rzadko uczniowie po nie sięgają. Wnika to głównie z braku satysfakcjonującej rozrywki oraz często nieadekwatnym poziomem wyzwania. Według portalu o nowoczesnej edukacji *edunews.pl*:

„*Pojęcie edurozrywki często bywa nadużywane, stosowane do określenia produkcji, które tak naprawdę cechują się znikomym poziomem faktycznej zabawy, z kolei wartość edukacyjna jest albo zbanalizowana albo zbytnio skomplikowana i nie wkomponowana w rozgrywkę. Odnalezienie równowagi pomiędzy stopniem rozrywki a poziomem oferowanej wartości edukacyjnej to jest częsty problem, z którym spotykają się producenci gier edukacyjnych. Zachowanie tych obydwu elementów na wysokim poziomie jednocześnie, jest dużym wyzwaniem*[14].”

Tymczasem gry z gatunku cRPG (ang. *computer Role Playing Game*), pokazują, że tego typu rozrywka mogą być bardzo dobrym narzędziem do nauki. Często wymagają one przyswojenia sporej ilości wiedzy. Fikcyjny świat gry bywa to dodatkową zachętą do jej zgłębiania. Według portalu o nowoczesnej edukacji *edunews.pl:*

„*Niektóre, bardziej złożone gry wymagają od użytkowników specjalizacji w konkretnej dziedzinie, czy określonej klasie postaci, którą sterują. Gry takie oferują bardzo rozbudowane i skomplikowane ścieżki rozwoju i strategie prowadzenia rozgrywki i dzieje się to w kontekstach science – fiction*

albo fantasy. Często wymaga to przyswojenia ogromnej ilości informacji i opanowania wiedzy na dany temat. Entuzjaści danego gatunku czy tytułu z reguły chętnie zgłębiają tajniki wykreowanego świata i niezbędne informacje, a co ciekawe, w wielu przypadkach właśnie ten element jest szczególnie zachęcający, jest czynnikiem dającym przewagę nad innymi graczami – większa wiedza na temat gry pozwala osiągnąć lepsze w niej rezultaty[11].”

Można wysnuć przypuszczenie, z przytoczonych powyżej przesłanek, iż dobra konstrukcja scenariusza w grze i odpowiednie stopniowanie trudności, będzie sprzyjać zaangażowaniu graczy i spowoduje eksplorowanie danego zakresu wiedzy wymaganego do rozwiązania wyzwań w rozgrywce. Dobrym przykładem jest wydana w 1999 roku przez firmę *Ruske & Puhretmaier Edutainment*[15] gra edukacyjna o nazwie *Physicus*, w której fabuła opierała się o rozwiązywanie zagadek fizycznych, rozwiązanie ich ratowało świat przed zagładą. Jednocześnie gra oferowała wysoką dokładność przedstawionego świata. Jej główny bohater, młody chłopiec stawał w obliczu wielkiego zagrożenia pochodzącego z kosmosu. W Ziemię uderza bowiem gigantyczny meteoryt. W wyniku kolizji nasza planeta przestała obracać się wokół własnej osi. Na jednej z półkul trwa więc wieczna noc, na drugiej zaś wieczny dzień, i to bardzo gorący. Delikatny ekosystem globu został poważnie naruszony. Naukowcy stwierdzili na szczęście, że można odwrócić ten proces i ponownie wprawić Ziemię w ruch wirowy. Wystarczy tylko wygenerować dostatecznie dużo energii, by uruchomić skomplikowaną maszynę. Recenzenci z portalu *gry.onet.pl* wyrazili następującą opinię o tej grze:

„Gracz rozwiązywał więc zadania fizyczne z dziedziny optyki, mechaniki, akustyki, elektromagnetyzmu i termodynamiki. Podziwiał kolorowe animacje i czytał podpowiedzi, a przy okazji zwiedzał wirtualny świat. „Fizykus” został bowiem tak zaprogramowany, by jednocześnie bawić i uczyć. Zadanie to spełniał zaś na tyle dobrze, że w wielu europejskich krajach wspiął się na szczyty list najlepiej sprzedających się programów[16].”

Jest ona doskonałym przykładem gry edukacyjnej, wskazuje na duże możliwości wykorzystywania dobrych konstrukcyjnie gier, łączących edukację i zabawę, które wspierają nauczanie przedmiotów ścisłych.

2 Cele pracy

Na wybór tematu pracy inżynierskiej wpłynęły przede wszystkim zainteresowania autora szeroką dziedziną gier komputerowych, a w szczególności problematyka wykorzystania ich w edukacji. Taka forma realizacji pracy dyplomowej jest okazją do sprawdzenia posiadanych umiejętności w praktyce, jednocześnie umożliwia zdobycie nowego doświadczenia w zakresie tworzenia gier komputerowych.

Głównym celem niniejszej pracy inżynierskiej jest opracowanie autorskiej gry komputerowej i sprawdzenie możliwości przedstawienia układów mechanicznych maszyn prostych w interaktywnej formie. Aspekt interaktywnego charakteru prezentowanych modeli prostych maszyn mechanicznych pozwala jednocześnie ocenić, czy opracowywana w projekcie gra jest wystarczająca, aby odbiorca – uczeń traktował ją jako operacyjne pogłębienie wiedzy teoretycznej przekazanej w ramach zajęć lekcyjnych.

Treść pracy dyplomowej przybliża również proces powstawania gier komputerowych. Porusza dylematy i problemy, przed którymi staje większość początkujących programistów oraz przedstawia ich przykładowe rozwiązania.

2.1 Zakres pracy

Niniejsza praca podzielona została na cztery zasadnicze części. Pierwsza część zawiera rozdział pierwszy i drugi. Pierwszy dokonuje przeglądu dostępnych na rynku gotowych rozwiązań i gier w zastosowaniach edukacyjnych. Wyjaśnione są w nim podstawowe definicje dla wprowadzenia czytelnika w tematykę pracy. Drugi omawia cele i zakres pracy. Następną, główną część pracy stanowią rozdziały 3 - 6. Rozdział 3 zawiera omówienie użytego silnika do gier. Rozdział 4 opisuje wykorzystane modele matematyczne. Rozdział 5 przedstawia sposób implementacji poszczególnych maszyn prostych w ramach gry oraz opisuje rozwiązane problemy wraz z ich przyczynami. Rozdział 6 to podsumowanie, wnioski oraz prawdopodobne sposoby na dalszy rozwój projektu w przyszłości. Część trzecia to rozdział 7. Stanowi on spis bibliografii. Czwartą częścią pracy jest realizowana aplikacja oraz jej kod źródłowy znajdujący na się na płycie CD. Wszystkie rysunki znajdujące się w pracy są oznaczone dodatkowym, krótkim opisem. Jeżeli ilustracja nie jest wykonana przez autora pracy, informacja o jej pochodzeniu została umieszczona na końcu jej opisu.

3 Tworzenie aplikacji (gry) – silnik gry

Praktyczną częścią niniejszej pracy inżynierskiej jest projekt gry o charakterze edukacyjnym pozwalającej na interakcję z wybranymi modelami maszyn prostych pod roboczym tytułem Mechanicus, wspierającej uczenie się fizyki. Jednym z pierwszych dylematów, przed którym staje autor pracy jest wybór silnika gry. Ważną przesłanką wyboru silnika jest kryterium dostępności dla jak największej liczby osób. Umożliwia to silnik obsługiwany przez wiele systemów operacyjnych. Kolejną podjętą decyzją co do kształtu gry jest wybór rodzaju grafiki. Przyjęto, że aplikacja będzie opierać się na grafice dwuwymiarowej. Głównym powodem jest to, iż idąca w parze z nią dwuwymiarowa implementacja fizyki wymaga mniej zasobów sprzętowych niż trójwymiarowa, co również zwiększa dostępności gry. Założono, iż realizacja gry nie będzie generować dodatkowych kosztów finansowych, co ułatwi ewentualne rozpowszechnienie. W sferze merytorycznej głównym motywem realizacji gry jest fizyka, a w szczególności maszyny proste do których autor wykorzysta mechanizmy wbudowane w silnik. Do stworzenia gry wybrano Godot Engine wraz z towarzyszącymi mu narzędziami.

Autor zdecydował się na ten silnik kierując się następującymi czynnikami:

- dostępność dla każdego jako oprogramowanie Open Source,
- kompatybilność projektu z przyszłymi wersjami systemów operacyjnych,
- łatwy dostęp do najnowszej wersji kodu i dokumentacji poprzez portal Github,
- możliwość skompilowania pod platformy: Windows, Mac OS X, Linux, HTML5, Android, BlackBerry,
- edytor wbudowany w silnik dający możliwość tworzenia uczniom i nauczycielom własnych poziomów,
- głęboki stopień zintegrowania ze sobą poszczególnych narzędzi (daje możliwości zaprogramowania przebiegu jakiegoś wydarzenia poprzez animacje oraz dostęp do zmiennych w skryptach z poziomu edytora graficznego scen),
- gra napisana w tym silniku nie wymaga od gracza instalacji dodatkowych programów jak np.: Java, Mono, .NET czy Flash,
- wbudowany silnik fizyczny wspiera wykrywanie kolizji.

3.1 Silnik gry - Godot

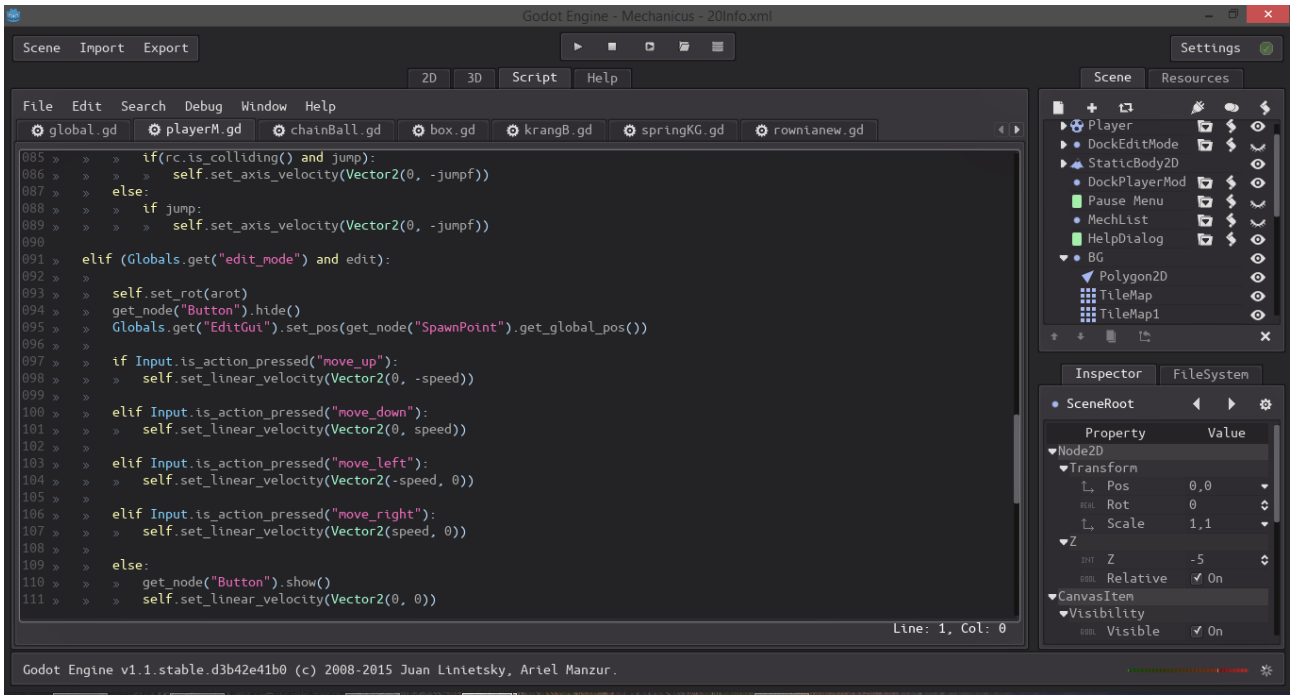
Silnik wybrany do tego projektu był rozwijany przez firmię Okam Studio w ciągu 11 lat. Dla użytkowników zewnętrznych jest dostępny jako oprogramowywanie Open Source od 7 kwietnia roku 2013, kiedy to opublikowano kod na portalu Github (https://github.com/okamstudio/godot), natomiast 15 grudnia 2014 opublikowano pierwszą wersję stabilną na oficjalnej stronie silnika (http://www.godotengine.org). Firma wykorzystała go tworząc głównie na urządzenia mobilne gry komputerowe na takie jak np.:

- *„El Asombroso Show Zamba”*,
- *„Dog Mendonça & Pizza Boy”*,
- *„Anthill”*,
- *„Running Nose”*,
- *„Project Carnival”*.

Jednak żadna z nich nie jest grą edukacyjną.

3.2 Skryptowe programowanie

W niniejszym podrozdziale opisane są możliwości programowania, przy pomocy skryptów w silniku Godot.



Rys. 1: Fragment kodu sterującego postacią gracza – okno edytora

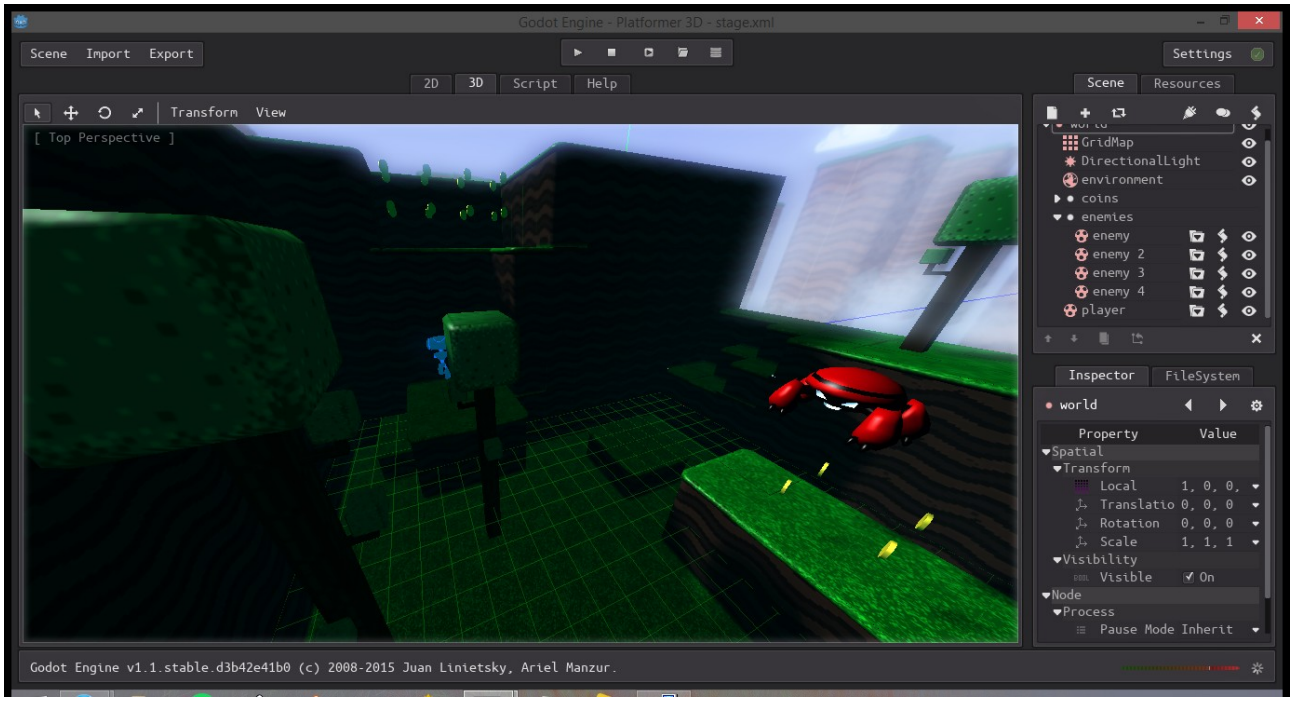
Godot posiada swój własny język skryptowy zwany GDScript, który jest podobny do języka Python. Jest językiem wysoko poziomowym i dynamicznie typowanym. Posiada również wbudowany edytor z automatycznymi wcięciami, podświetlaniem tekstu, auto uzupełnianiem kodu (od wersji 1.1) oraz debugger, który wspiera tzw. breakpointy i wykonywanie kodu krok po kroku.

Najważniejsze równice pomiędzy GDScript a Pythonem to:

- każdą zmienną trzeba zdeklarować używają słowa kluczowego **var**
- każdy skrypt jest klasą rozszerzającą, pochodną klasy Node i zaczyna się od słowa kluczowego **extends**
- zmienne i funkcje własne danej klasy można poprzedzić słowem kluczowym **self** (w Pythonie trzeba)
- GDScript posiada sporo dodatkowych typów wbudowanych przydatnych w programowaniu gier

3.3 Renderowanie

Podrozdział o możliwości wykorzystania trójwymiarowej grafiki w tym silniku.

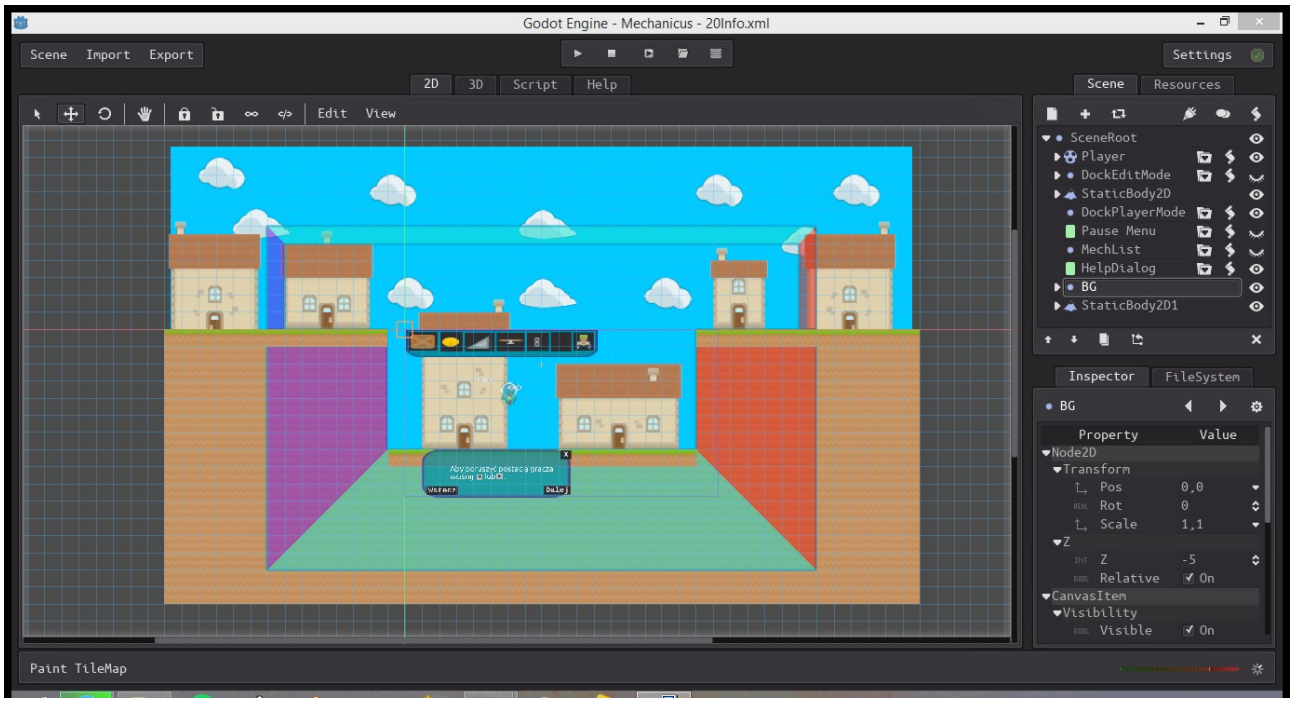


Rys. 2: Przykładowa trójwymiarowa scena – okno edytora

Silnik graficzny Godota na każdej platformie wykorzystuje OpenGL ES 2.0, który jest podzbiorem biblotek OpenGL używanym głównie w mobilnych systemach operacyjnych. Dzięki niemu Godot wspiera przezroczystość, tzw mapy normalnych (ang. *normal mapping*), poziom szczegółowości modeli 3D, dynamicznie cieniowanie korzystając z map cieni (ang. *shadow maps*) oraz efekty takie jak:FXAA, bloom, DOF, HDR, korekcja gamma i mgła.

3.4 Gry z grafiką 2D

Podrozdział o jednym z mechanizmów użytych w projekcie, czyli trybie tworzenia gier dwuwymiarowych.



Rys. 3: Dwuwymiarowa scena z omawianej gry – okno edytora

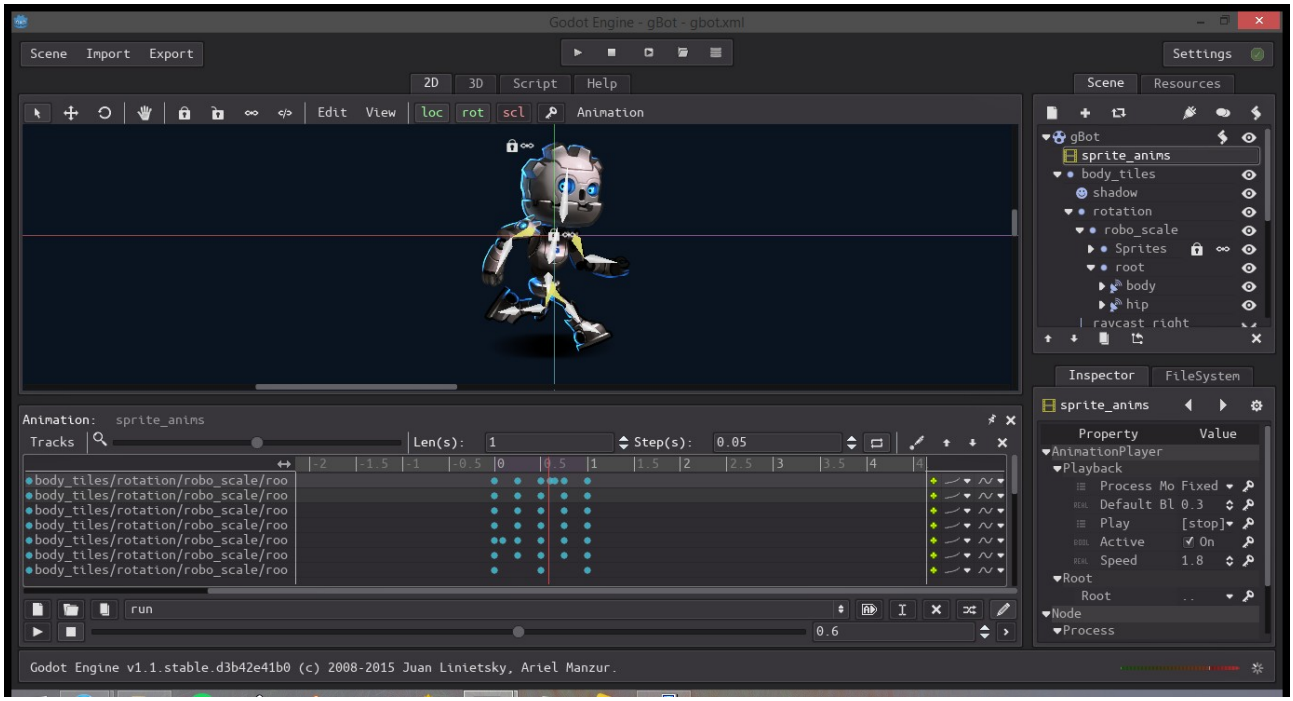
Silnik Godot posiada wbudowany edytor gier dwuwymiarowych. Zawiera on moduły do tworzenia: interfejsu graficznego użytkownika (ang. *graphical user interface*), duszków (ang. *sprites*), poligonów, animacji jak również silnik fizyki, system efektów cząsteczkowych, podstawowe cieniowanie i oświetlenie. Oprócz tego popularne i wykorzystywane w wielu grach: przewijane tła (ang. *parallax scrolling*), konstruowanie poziomów z tzw. kafelków (ang. *tile sets*), zaawansowany system cieniowania (ang. shaders).

Istnieje także możliwość mieszania tego trybu z trybem 3D poprzez wykorzystanie elementu Viewport.

Godot posiada dedykowany silnik dla gier 2D podczas gdy np.: popularny silnik **Unity 3D**, posiada jedynie tryb tzw. fałszywego 2D, gdzie dwuwymiarowe grafiki są nadkładane jako tekstury na płaskie obiekty w trójwymiarowej przestrzeni.

3.5 Animacja

Podrozdział opisuje system animacji dostępny w Godot.



Rys. 4: Przykładowa animacja – okno edytora

Godot posiada wyszukany system animacji z pełnym wsparciem dla: edycji, animacji szkieletowej, drzew animacji, tzw. cutscenek w czasie rzeczywistym, przejść po między animacjami (ang. *blending*), morfingu, wywoływaniem funkcji, zmiany w czasie każdej z dostępnych właściwości danego obiektu.

Każda w właściwości danego obiektu ma własną ścieżkę animacji. Dla każdej ścieżki możemy ustawić aby była ciągła (komputer wylicza jej wartość w czasie pomiędzy kolejnymi klatkami kluczowymi), nieciągła (ang. *discrete*) (brak wyliczonych wartości pomiędzy kolejnymi klatkami kluczowymi). Oraz algorytm wyliczania tych wartości: linowy (domyślny), interpolacyjny (ang. *cubic*), algorytm najbliższego sąsiada (ang. *nearest neighbour*).

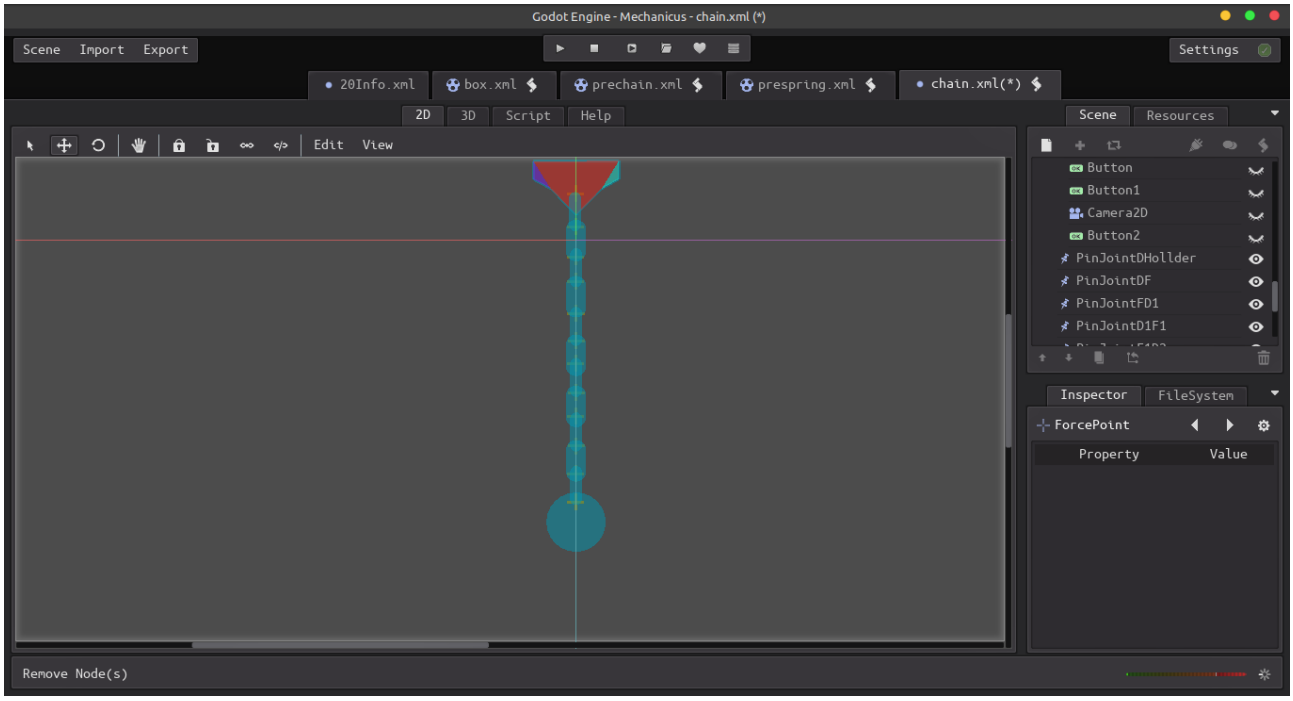
Dla każdej animacji możemy ustawić prędkość wykonywania, długość, czy ma się pętlować oraz auto odtwarzanie po wczytaniu sceny.

Animacja szkieletowa posiada system kinematyki odwrotnej.

Gotową animację można wyeksportować i użyć w innej senie na innym obiekcie pod warunkiem zachowania nazw obiektów zgodnych ze sceną z której pierwotnie wyeksportowaliśmy animację.

3.6 Silnik Fizyki

Podrozdział o jednym z mechanizmów użytych w projekcie, czyli silnika fizyki wbudowanego w Godot.



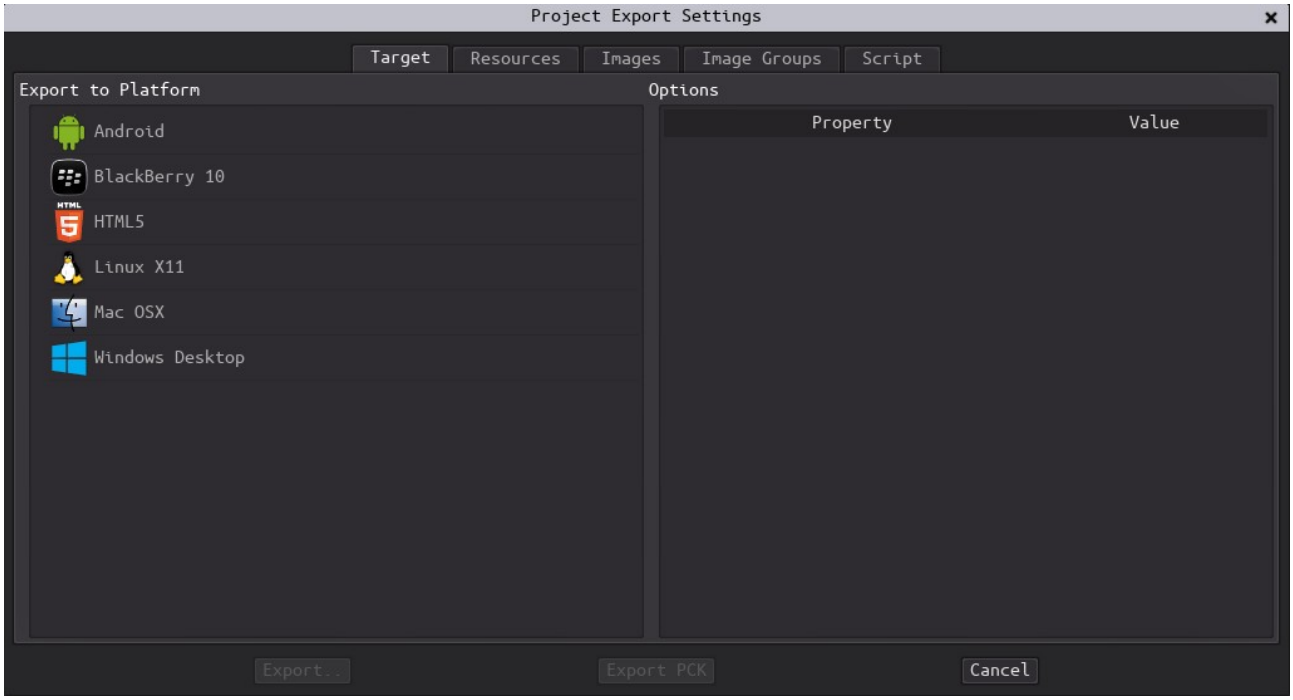
Rys. 5: Obiekt chain.xml pokazujący swą budowę fizyczną

Silnik Godot posiada również własną wbudowaną symulację zjawisk fizycznych (tzw. silnik fizyczny) dla trybu 2D jak i 3D. Wspiera on wykrywanie kolizji, ciała sztywne (ang. *rigid body)*, ciała statyczne (ang. *static body)*, postacie, pojazdy, czujnik kolizji (ang. *raycasts*) oraz różnego rodzaju łączenia (ang. *Joints*).

Wykrywanie kolizji jest realizowane poprzez nadanie ciału odpowiedniego kształtu kolizji (ang. *collision shape*). Wykonuje się to dodając do struktury obiektu element CollisionShape2D (zawiera on zbiór podstawowych kształtów z których możemy wybrać najbardziej pasujący) lub element CollisionPolygon2D (gdy mamy bardziej skomplikowany obiekt). Poszczególne obiekty fizyczne możemy łączyć wykorzystując obiekty pochodne od Joint2D:

- PinJoint2D – połączenie osiowe,
- DampedSpringJonit2D – połączenie sprężynowe,
- GrooveJoint2D – połączenie rowkowe.

3.7 Wspierane platformy



Rys. 6: Okno eksportu aplikacji

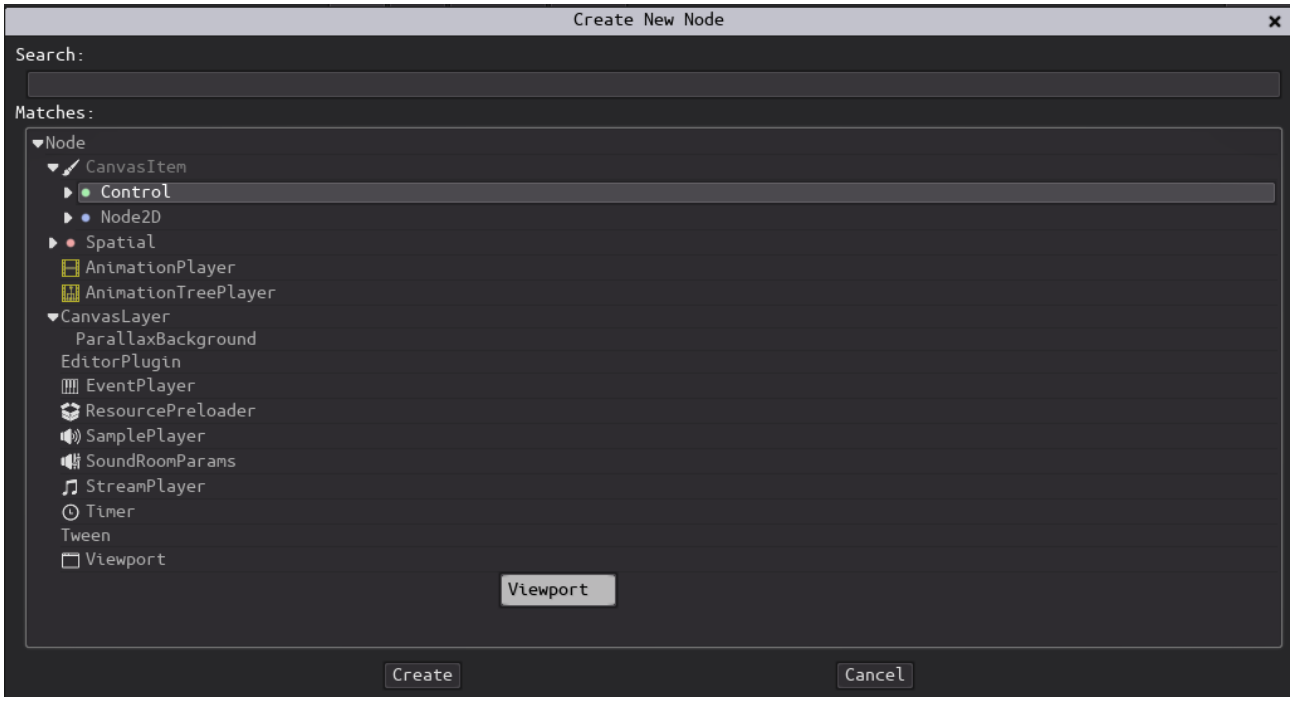
Godot wspiera kompilację na wiele platform. Z poziomu projektu programista może wybrać platformę docelową, obecnie obsługiwane są: Linux, Mac OS X, Windows, Android, BlackBerry 10, HTML5. Godot pozwala również na ustawianie specyficznych ustawień dla każdej ze wspieranej platform jak np.: kompresję tekstur czy rozdzielczość.

Dzięki silnikowi Godot programista może dodatkowo:

- obsługiwać trójwymiarowe tekstury,
- generować wykresy osiągów,
- wstępnie szacować rozmieszczanie światła tzw. Light baking,
- obsługiwać wielowątkowość,
- odtwarzać wideo korzystającego z kodeka Theora,
- odtwarzać plików audio w formacie wav oraz ogg,
- używać systemu cząsteczek i wtyczek.

3.8 Rodzaje obiektów silnika

Podstawowym „budulcem” każdej Sceny są obiekty zwane Node.



Rys. 7: Okno dodawania nowego obiektu do sceny

Wszystkie one pochodzą od klasy Node. Wypisuje tylko te pochodne, które zostały użyte w grze. Dzielą się na parę główny grup:

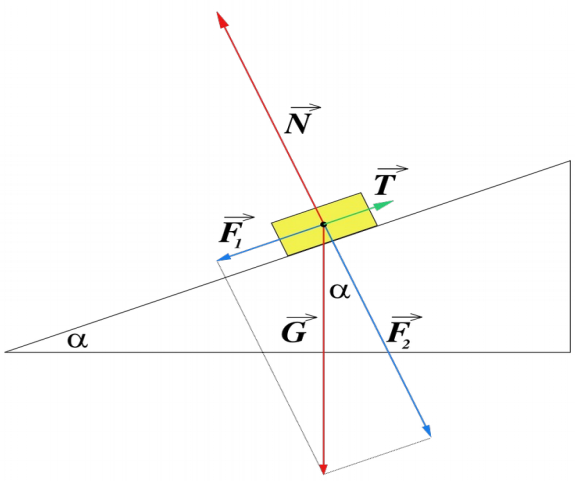
- CanvasItem:
  - Control – Interfejs użytkownika (oznaczone zielonym kolorem):
    - BaseButton:
      - Button – Przycisk,
    - Range:
      - Label – Napis,
      - LineEdit – jednoliniowe pole tekstowe,
      - Slider:
        - HSlider – suwak poziomy,

        - Popup:
            - PopupPanel – wyskakując panel,
        - RichTextLabel – wielolinijkowe pole tekstowe obsługujące tzw, BBCode,
    - Node2D – Do tworzenia gier 2D (oznaczone niebieskim kolorem):
        - Sprite – grafika 2D,
        - Postion2D – służy do ustalania pozycji ic
        - CollisionObject2D:
            - PhysicsBody2D:
                - RigidBody2D – ciało sztywne,
                - StaticBody2D – ciało stałe,
        - CollisionShape2D,
        - CollisionPolygon2D,
        - RayCast2D,
        - Polygon2D,
        - Camera2D,
        - Joint2D:
            - PinJoint2D – połączenie osiowe,
            - DampedSpringJonit2D – połączenie sprężynowe,
            - GrooveJoint2D – połączenie rowkowe,
        - TileMap – część poziomu skonstruowana z tzw. kafelków (ang. *tiles*)
- Spatial – Do tworzenia gier 3D (oznaczone różowym kolorem),
- Animation – do zarządzania i tworzenia animacji,
- pozostałe – do zarządzie zasobami i zdarzeniami oraz ViewPort (oznaczone szarym oraz białym kolorem).

4 Modele matematyczne – zaimplementowane w aplikacji (grze)

W tym podrozdziale opisano modele matematyczne, które autor uwzględnia przy tworzeniu oraz testowaniu implementacji poszczególnych maszyn. Ponieważ gra ma z założenia być edukacyjna, autor bierze pod uwagę treści z podręcznika do fizyki dla szkół ponadgimnazjalnych.

4.1 Równia pochyła



Rys. 9: Równia pochyła źródło wikipedia.org

Najważniejszy jest rozkład sił. Od niego zależy, czy umieszczone na równi ciało będzie się zsuwać i z jakim przyspieszeniem, czy pozostanie w spoczynku, w wyniku równoważenia się siły F_1 i T.

Mamy dwa rodzaje tarcia ciała znajdującego się na równi:

- tarcie statyczne, gdy ciało stoi w miejscu,
- tarcie kinetyczne, gdy ciało porusza się wzdłuż powierzchni pochyłej równi.

Są one wyliczane z wzoru:

$$T = f N = f mg \cos \alpha$$

gdzie: *T – tarcie, f - współczynnik tarcia, N - siła nacisku, m – masa ciał, g - ziemskie przyśpieszenie, α – kąt nachylenia równi.*

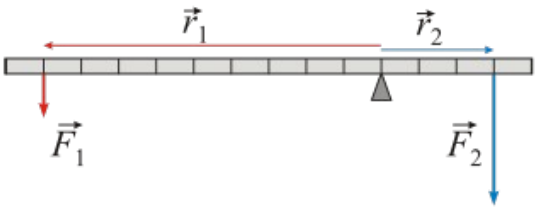
Potrzeba jeszcze wzoru na przyśpieszenie ciała suwającego się z równi:

$$a = g(\sin \alpha - f \cos \alpha)$$

gdzie: *a - przyśpieszenie, f - współczynnik tarcia, α – kąt nachylenia równi*

4.2 Dźwignia dwustronna

Dla dźwigni dwuramiennej siły działają po przeciwnych stronach osi obrotu.



Rys. 11: Dźwignia dwustronna źródło wikipedia.org

Dźwignia pozostaje w równowadze, gdy wypadkowy moment sił do niej przyłożonych wynosi 0. Co można zapisać równaniem:

$$F_1 r_1 + F_2 r_2 = 0$$

gdzie: F_1 – *siła nacisku na pierwsze ramie dźwigni,* r_1 *– długość pierwszego ramienia dźwigni,* F_2 – *siła nacisku na drugie ramie dźwigni,* r_2 *– długość drugiego ramienia dźwigni.*

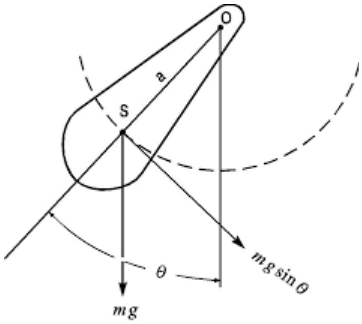
Można to przekształcić do wzoru na tzw. *przełożenie dźwigni* :

$$\frac{F_2}{F_1} = \frac{r_1}{r_2}$$

gdzie: F_1 – *siła nacisku na pierwsze ramie dźwigni,* r_1 *– długość pierwszego ramienia dźwigni,* F_2 – *siła nacisku na drugie ramie dźwigni,* r_2 *– długość drugiego ramienia dźwigni.*

4.3 Wahadło fizyczne

Wahadło fizyczne nie jest podręcznikową maszyna prostą, ale przedstawia elementarne zjawiska fizyczne. Wahadło fizyczne to bryła sztywna zawieszona na poziomym odcinku. Jest to bardziej realistyczna forma wahadła matematycznego.



Rys. 12: Wahadło fizyczne

Jest to bardziej realistyczna forma wahadła matematycznego. Gdy wahadło zostanie wychylone z położenia równowagi, wówczas działa na nie moment siły:

$$M = mga\sin\theta$$

Równanie ruchu wahadła wyraża wzór:

$$\frac{a^2\theta\ (t)}{at^2} + \frac{mga}{I}\sin\theta\ (t) = 0$$

Aby zobaczyć jego zależność względem ruchu wahadła matematycznego, wprowadza się *długość zredukowaną* wahadła fizycznego:

$$l_f = \frac{I}{ma}$$

Tak zmienione równanie ruchu wahadła fizycznego ma taką samą formę jak równanie ruchu dla wahadła matematycznego, oznacza to, że wszystkie twierdzenia dotyczące ruchu wahadła fizycznego są zgodne z wnioskami dotyczącymi wahadła matematycznego.

Zakładając takie same warunki dla wahadła fizycznego jak dla wahadła matematycznego, tzn. przyjmując, że jest masą punktową zawieszoną na nieważkiej nici, otrzymamy następujące wielkości:

$$I = ml^2$$
$$a = l$$

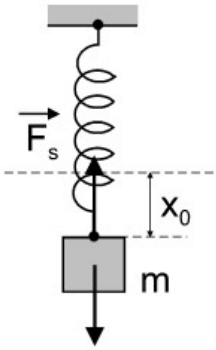
Po podstawieniu ich do równań wahadła fizycznego uzyskuje się równania ruchu jak dla wahadła matematycznego, dlatego też wahadło matematyczne może być traktowane jako szczególny przypadek wahadła fizycznego.

Gdzie:

- a – odległość od punktu zawieszenia do środka ciężkości bryły,
- g – przyspieszenie ziemskie,
- l – moment bezwładności wahadła względem osi obrotu,
- m – masa wahadła,
- l_f – długość zredukowana wahadła fizycznego.

4.4 Masa na sprężynie

Masa na końcu sprężyny jest ciągnięta raz w dół, a raz w górę przez siłę sprężystości. Wartość tej siły jest wprost proporcjonalna do wydłużenia sprężyny x.



Wydłużanie sprężyny jest równe wychyleniu ciała, czyli w tym przypadku odważnika, z położenia równowagi co daje nam wzór:

$$F = -k \cdot x$$

gdzie: *F – siła sprężystości, k – współczynnik sprężystości, x – wydłużenie sprężyny.*

Siła sprężystości dąży do przywrócenie pierwotnej długości sprężyny. Odpowiednio maleje i rośnie wraz z wartością wydłużenia sprężyny.

5 Realizacja aplikacji

W niniejszym rozdziale zostaną zaprezentowane podstawowe elementy silnika, które wykorzystano do zrealizowania gry oraz skonstruowania poszczególnych maszyn, postaci sterowanej przez gracza oraz interfejsu. Rozdział ten stanowi jednocześnie instrukcję do aplikacji, będącej przedmiotem tej pracy.

Nawiązując do typowych rozwiązań z gier użytkownik rozpoczyna rozgrywkę uruchamiając aplikację. Po włączeniu ma dostęp do poziomu testowego przedstawiającego rozgrywkę. Interakcja rozpoczyna się od ruchu ludzikiem „Mechanikiem” za pomocą klawiatury lub umieszczenia wybranej maszyny na planszy z menu znajdującego się nad „Mechanikiem”. Wybór maszyny dokonywany jest przez kliknięcie lewym przyciskiem myszy.

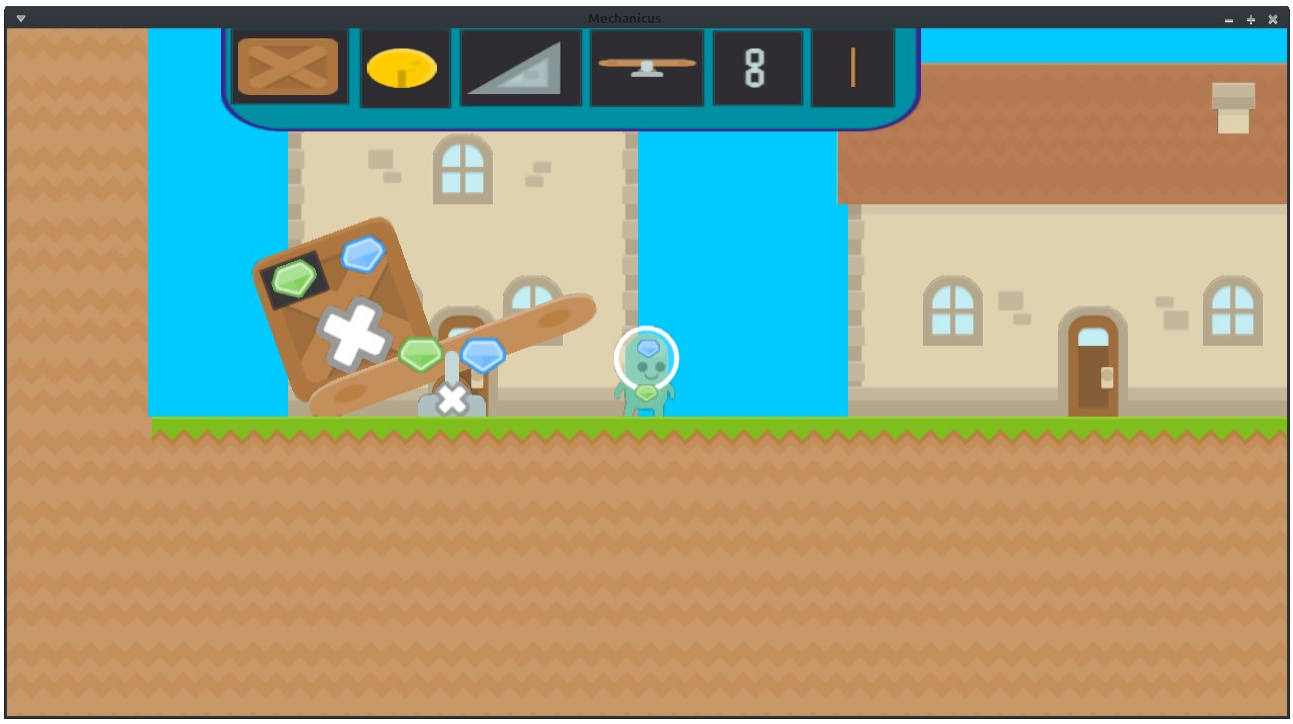
5.1 Tryby gry

Użytkownik ma do wyboru trzy tryby interakcji z maszynami – każdy tryb przewiduje poznanie zagadnień fizycznych z zakresu kinematyki na różnych poziomach zaangażowania. Każdy z nich realizuje inny stopień orientacji, począwszy od prostej interakcji a skończywszy na eksperymentowaniu:

- **Player** – tryb umożliwiający wchodzenie w interakcję z maszynami oraz dostęp do interfejsu wyboru maszyny do umieszczenia w rozgrywce;
- **Info** – tryb w którym gracz widzi zmienne właściwości fizyczne przeliczane w czasie rzeczywistym podczas rozgrywki;
- **Edycji** – tryb w którym gracz może przesuwać dowolnie wybraną maszyną oraz zmieniać jej wybrane właściwości fizyczne.

5.1.1 Tryb Player

Tryb Player jest domyślny i aktywuje się po uruchomieniu gry. W trybie tym gracz może wchodzić w interakcję z obiektami swoją postacią poprzez ruch na boki oraz skoki korzystając z klawiatury (strzałki i spacja). Za pomocą myszki może wybrać z paska nad graczem maszynę, którą chce umieścić na planszy. W przypadku niektórych z nich ustawić wstępne parametry.



Rys. 14: Gra w trybie Player

Przełączanie pomiędzy trybami **Player** a **Edycja** jest zrealizowane za pomocą zmiennej globalnej *edit_mode*, której zmiana wartości na *true* włącza odpowiednią część skryptu danej maszyny prostej dla trybu edycji, natomiast wartość *false* powoduje wyłącznie tej części kodu i powrót do trybu Player. Jest to zabezpieczanie przed przypadkową edycją kilku obiektów jednocześnie. Aby zmienić tryb **Player** na tryb **Edycji** należy kliknąć na zielony kryształ na wybranym obiekcie, z kolei zmiana na tryb **Info** odbywa się poprzez kliknięcie na niebieski kryształ.

Rys. 15: Podpowiedzi w grze

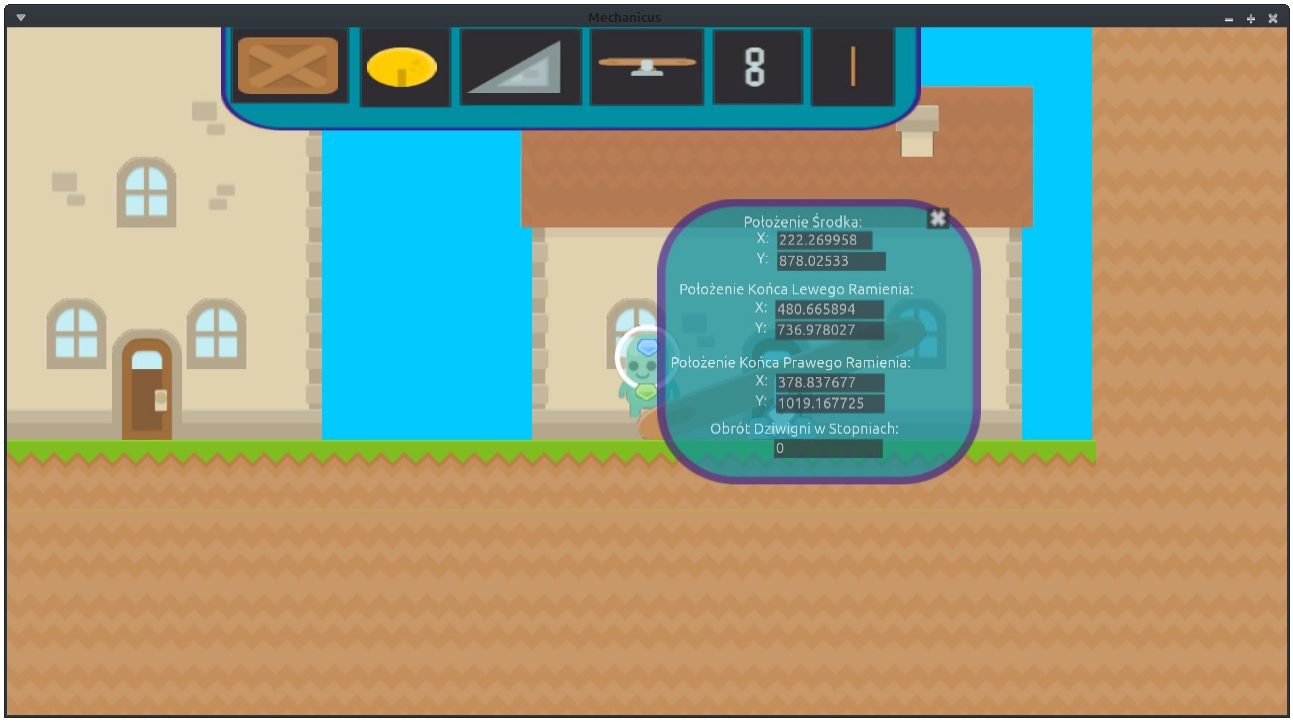
Gdy gracz uruchomi grę, lub włączy pomoc to to pojawi się seria podpowiedzi objaśniająca co może w grze zrobić i za pomocą czego. Podpowiedzi wyjaśniają graczowi jak sterować postacią, jak wstawić maszynę, itd.

Rys. 16: Lista Maszyn w grze

Lista dostępna jest z menu pauzy oraz w trzecim kroku pomocy. Każda maszyna ma opisane swoje właściwości fizyczne i wzory w Liście Maszyn. Opisy pochodzą z stron Wikipedii, gdyż był to najłatwiejszy sposób na umieszczanie ich w grze poprzez obiekt RichTextLabel, który obsługuje tzw. BBCode. Artykuły z Wikipedii łatwo skonwertować do tego kodu korzystając ze stron: https://tools.wmflabs.org/blogconverter/[29]. i http://www.garyshood.com/htmltobb/[30]..

5.1.2 Tryb Info

Rys. 19: Tryb Info włączony na równi pochyłej



Rys. 20: Dźwignia dwustronna w trybie Info

W trybie **Info** w oknach dialogowych obok wybranych przez gracza maszyn wyświetlają się następujące wartości fizyczne zmienne w czasie:

- Tarcie,
- Prędkość Linową,
- Prędkość Kątowa,
- Położenie.

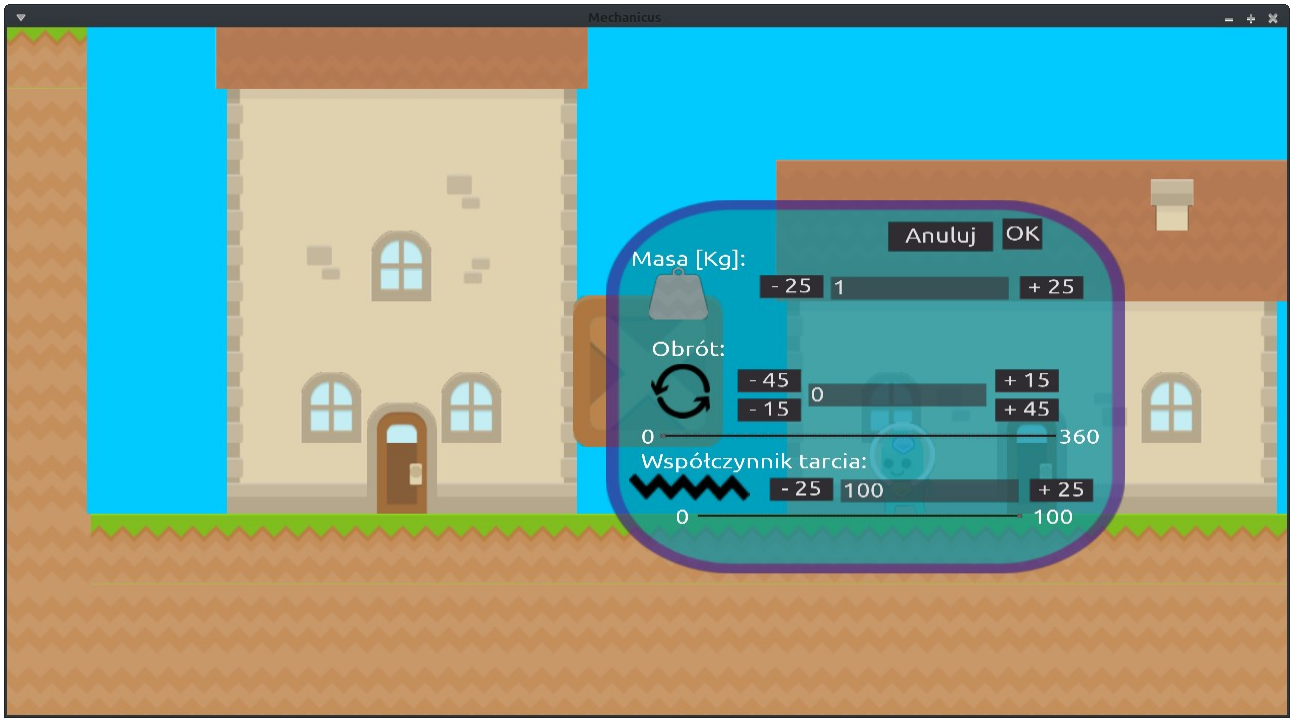
Wyjątkiem od tej reguły są dźwignie, którym tryb **Info** pokazuje:

- Położenie środka dźwigni,
- Położenie końca lewego ramienia,
- Położenie końca prawego ramienia,
- Obrót dźwigni względem środka.

5.1.3 Tryb Edycji

W trybie edycji możemy ustawić następujące parametry fizyczne większości maszyn prostych:

- Masa,
- Obrót,
- Współczynnik Tarcia,
- Położenie.



Rys. 21: Tryb edycji masy na podłożu z tarciem w grze

Takie maszyny jak masa na sprężynie i wahadło fizyczne posiadają dwa rodzaje Edycji:

1. Pierwsza polega na możliwości przemieszczania obiektu z uproszczonym systemem kolizji. Edycja ta aktywuje się po wybraniu maszyny z menu;

Druga aktywuje się po potwierdzeniu wstępnego położenia. Wówczas system kolizji zostaje zamieniony na bardziej zaawansowany i można edytować właściwości ciała przyczepionego do końca wahadła lub sprężyny.

5.1.4 Menu Pauzy

Rys. 22 : Menu Pauzy w grze

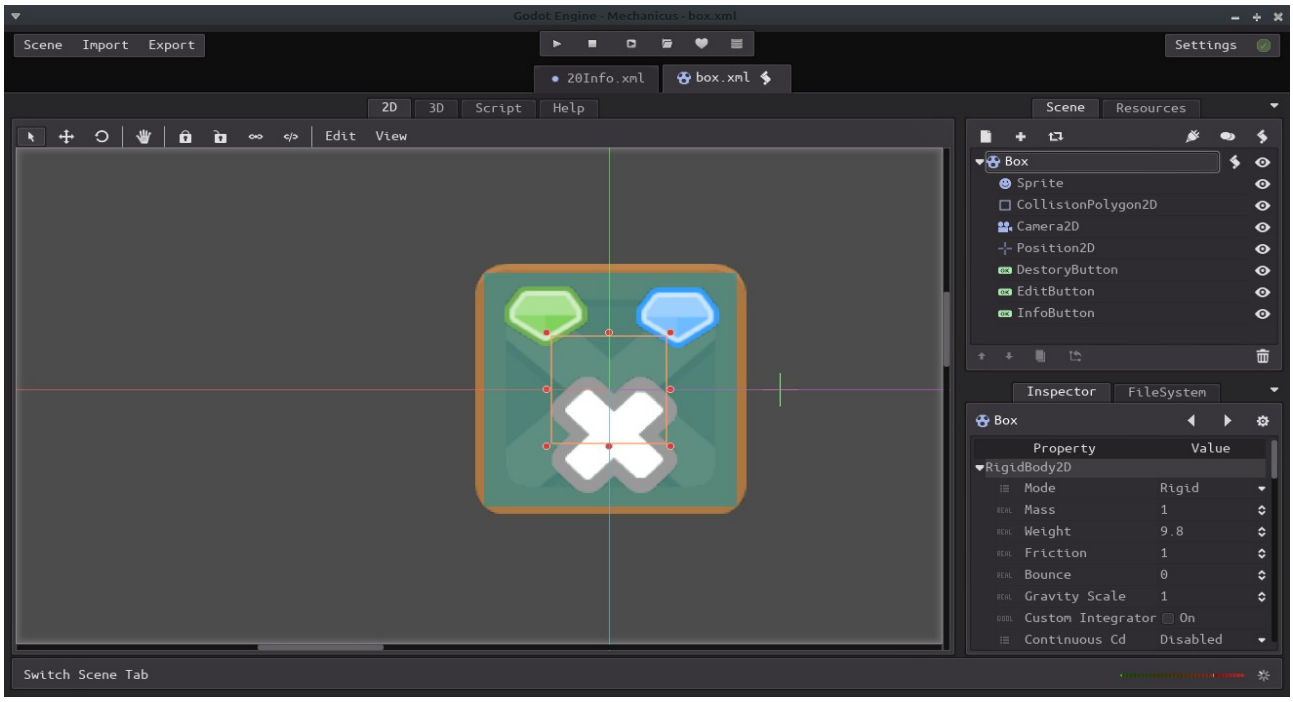
W każdym trybie gracz ma dostęp do Menu Pauzy, które wywołuje poprzez naciśnięcie na klawiaturze klawisza ***Esc***. Z jego poziomu możemy:

- kontynuować grę**,**
- ponownie uruchomić pomoc,
- otworzyć listę dostępnych maszyn,
- wyjść z gry.

5.2 Implementacja wybranych maszyn prostych

5.2.1 Masa na podłożu z tarciem

Autor przed zrealizowaniem wirtualnych wersji maszyn prostych stworzył implementację dodatkowego obiektu fizycznego będącego odwzorowaniem masy o kształcie sześcianu w trybie 2D. Obiekt ten był punktem wyjścia do implementacji maszyn oraz postaci „Mechanika”.

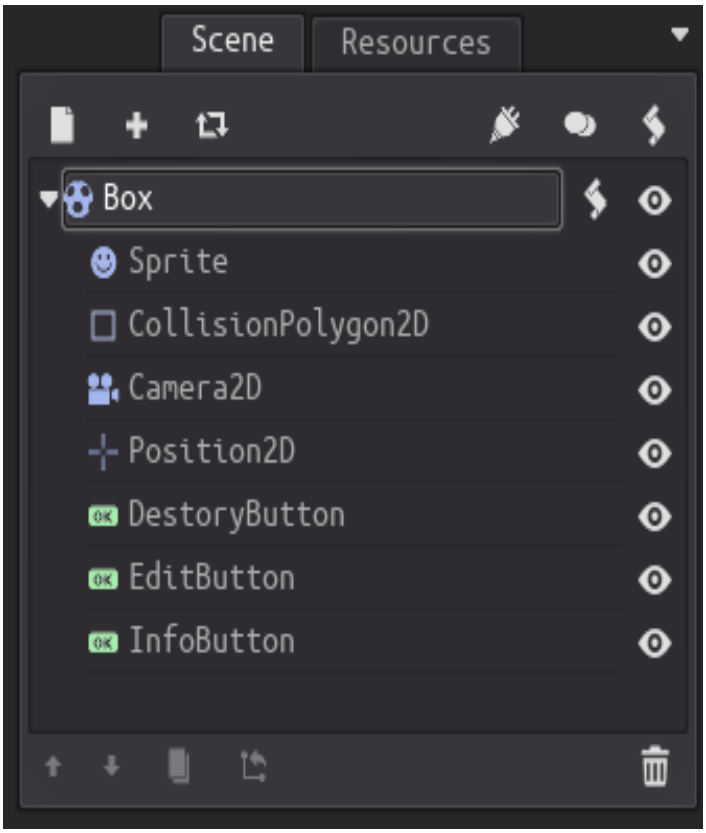


Rys. 23: Plik box.xml w edytorze

Implementację masy na podłożu z tarciem zawiera plik *box.xml*. Wszystkie obiekty fizyczne w grze są modyfikacją struktury jaką posiada obiekt *box.xml*:

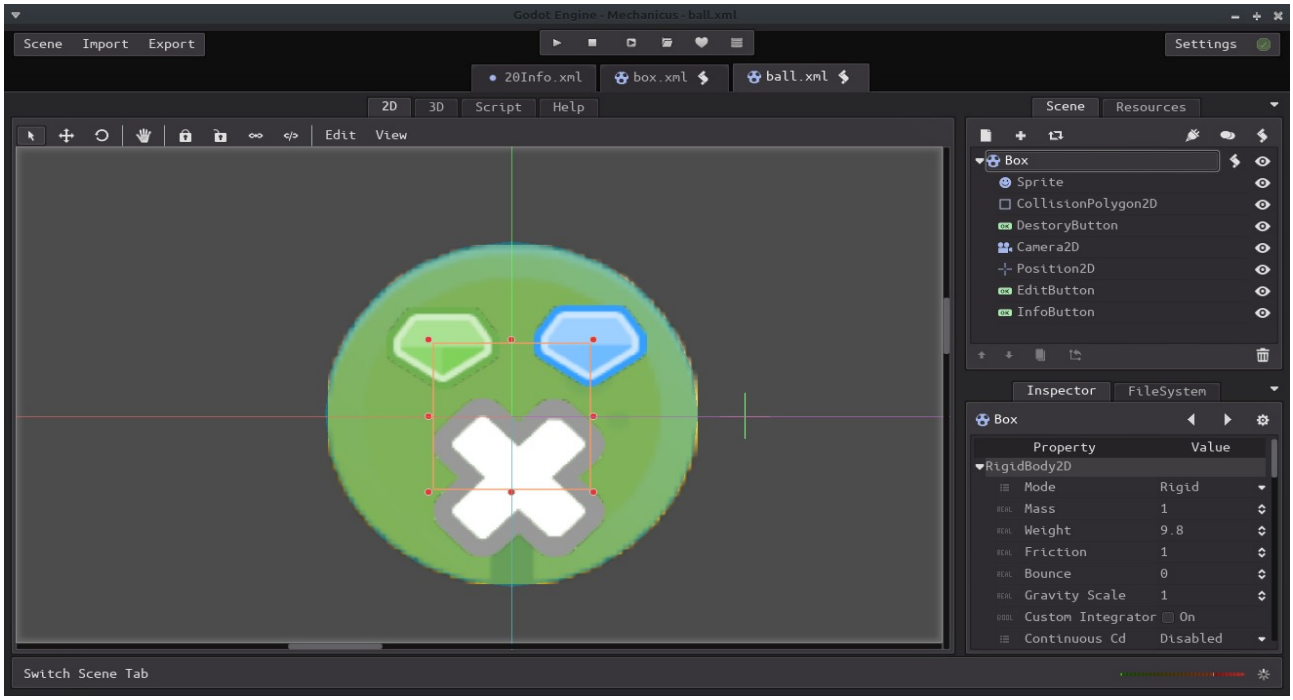
- RigidBody2D:
  - Sprite – służy za reprezentację graficzną obiektu,
  - CollisionPolygon2D lub CollisionShape2D - wykorzystany do zdefiniowania kształtu fizycznego obiektu i kolizji,
  - Camera2D – wykorzystany do śledzenia obiektu,
  - Postion2D – wykorzystany do przypina interfejsu do obiektów,

- DeleteButton (Button) - przycisk do usuwania obiektu, bądź ich grupy,
- EditButton (Button) - przycisk do przejścia w tryb edycji właściwości fizycznych obiektu,
- InfoButton (Button) – przycisk powodujący wyświetlenie właściwości fizycznych obiektu w czasie rzeczywistym.



Rys. 24: Struktura obiektu box.xml w edytorze

Dzięki dzięki zaawansowanym możliwościom silnika Godota, autor po stworzeniu modelu, musiał tylko dobrać odpowiedni kształt poprzez CollisionPolygon2D oraz w skrypcie (plik box.gd) opisać zachowanie się ciała trybie **Edycji**. Implementacja obiektu przebiegała podobnie w pozostałych maszynach uwzględnionych w aplikacji-grze.



Rys. 25: Plik ball.xml w edytorze

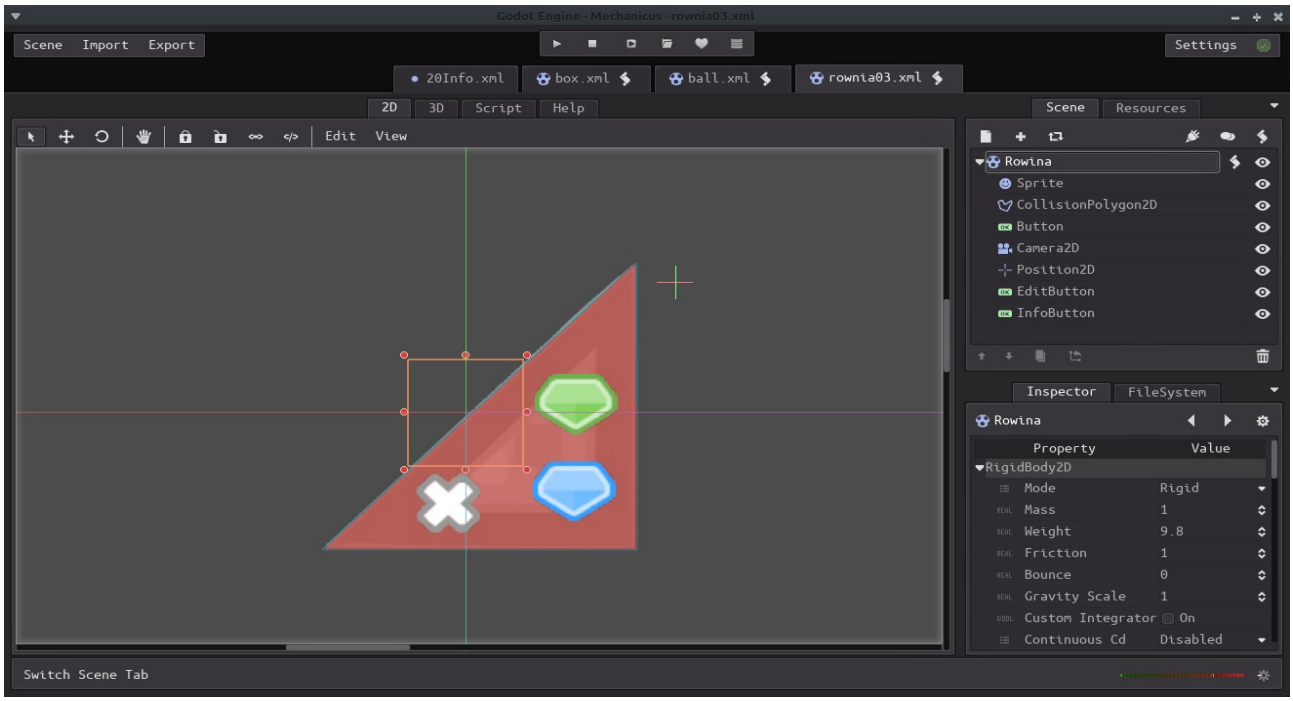
Dodatkowo autor zaimplementował kulę i dał możliwości edytowania właściwości fizycznych postaci gracza, aby zwiększyć możliwości eksperymentowania z umieszczanymi na planszy obiektami.

Aby stworzyć kulę wystarczyło tylko stworzyć kopię pliku *box.xml* opisującego implementację masy na podłożu z tarciem i zmienić w nim reprezentację graficzną oraz kształt CollisionBoxa.

Natomiast do postaci gracza wystarczyło dodać elementy z implementacji sześcianu i częściowo zmodyfikować skrypt.

5.2.1 Masa na równi pochyłej

Autorzy silnika Godot odradzają dynamicznie zmienianie rozmiarów obiektów korzystających z RigidBody2D lub StaticBody2D dlatego też implantacja równi pochyłej składa się z 4 plików.



Rys. 24: Plik rownia03.xml w edytorze

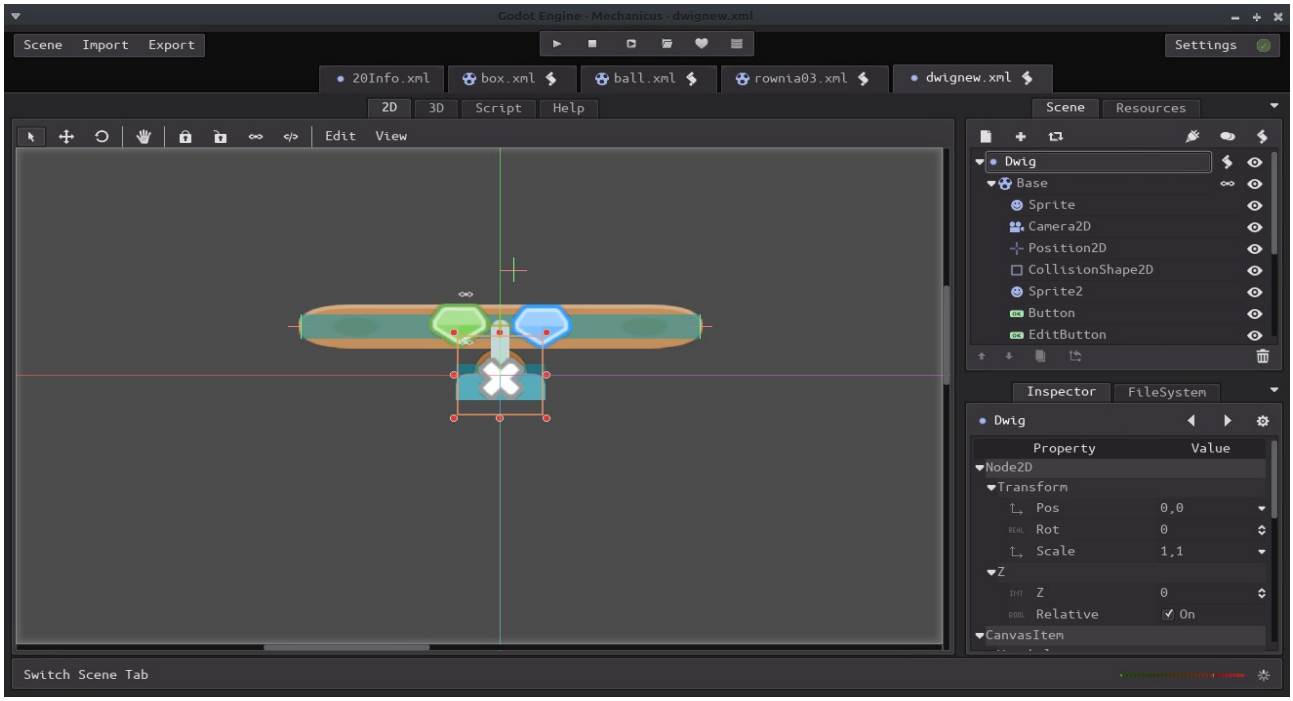
Rys. 26: Wybór rozmiaru równi pochyłej w grze

Wynika to z chęci umożliwienia użytkownikowi, zmiany stopnia nachylenia równi. aby to osiągnąć zostały stworzone 4 wersje równi, każda w innym rozmiarze:

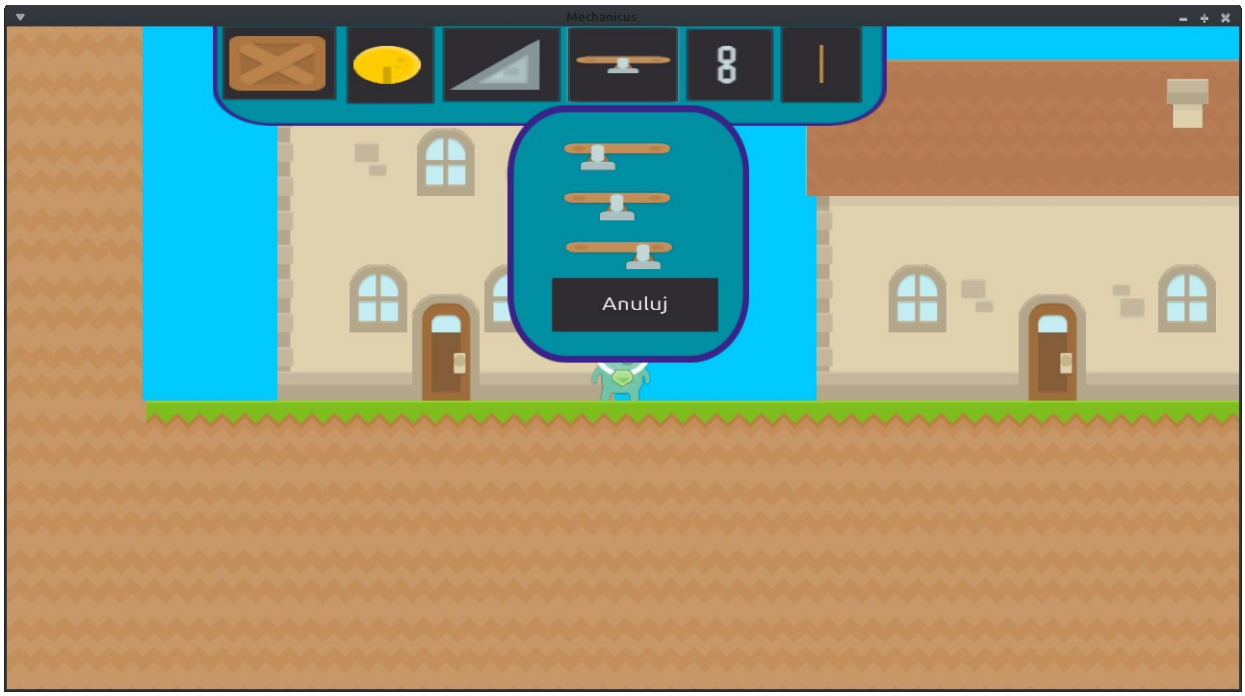
- rownia03.xml – 184 x 171 px
- rownia03x2.xml – 386 x 171 px
- rownia03y2.xml – 184 x 240 px
- rownia03x2y2.xml – 386 x 240 px

Podczas trybu **Player** po kliknięciu na przycisk równi pochyłej z menu nad graczem pokazuje się lista dostępnych rozmiarów. Aby wstawić równię należy wybrać jeden z nich.

5.2.2 Dźwignia dwustronna



Rys. 27: Plik dwignew.xml w edytorze



Rys. 28: Wybór miejsca przyłożenia dźwigni w grze

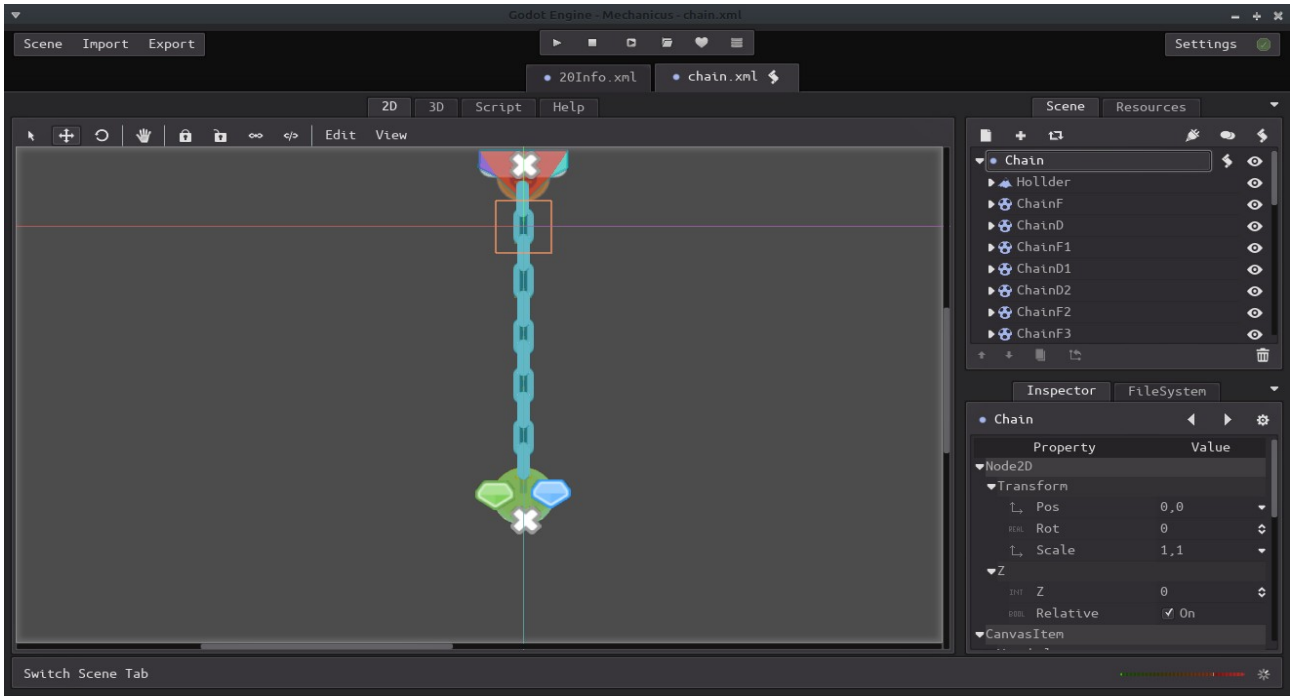
Dźwignia w przeciwieństwie do równi pochyłej czy sześcianu składa się z dwóch obiektów RigidBody2D. Aby połączyć ze sobą oba elementy, z których składa się dźwignia wykorzystano obiekt PinJonit2D. Została ona zaimplementowana za pomocą trzech plików.
Pliki te definiują dźwignie różniące się stosunkiem długości ramion względem siebie:

- dwignew.xml – stosunek ramion 1:1
- dwignews.xml – stosunek ramion 1:2
- dwignewe.xml – stosunek ramion 2:1

Gdy gracz kliknie przyciska dźwigni w trybie Player ukaże się menu pozwalając na wybór stosunku ramion.

5.2.3 Wahadło fizyczne

Pierwotnym zamiarem autora było stworzenie implementacji wahadła matematycznego, ale kazało się iż stopień zaawansowania silnika fizycznego powoduje, że bardzo prosto było stworzyć bardziej realistyczną wersję wahadła niż tzw. wahadło matematyczne, a mianowicie wahadło fizyczne.

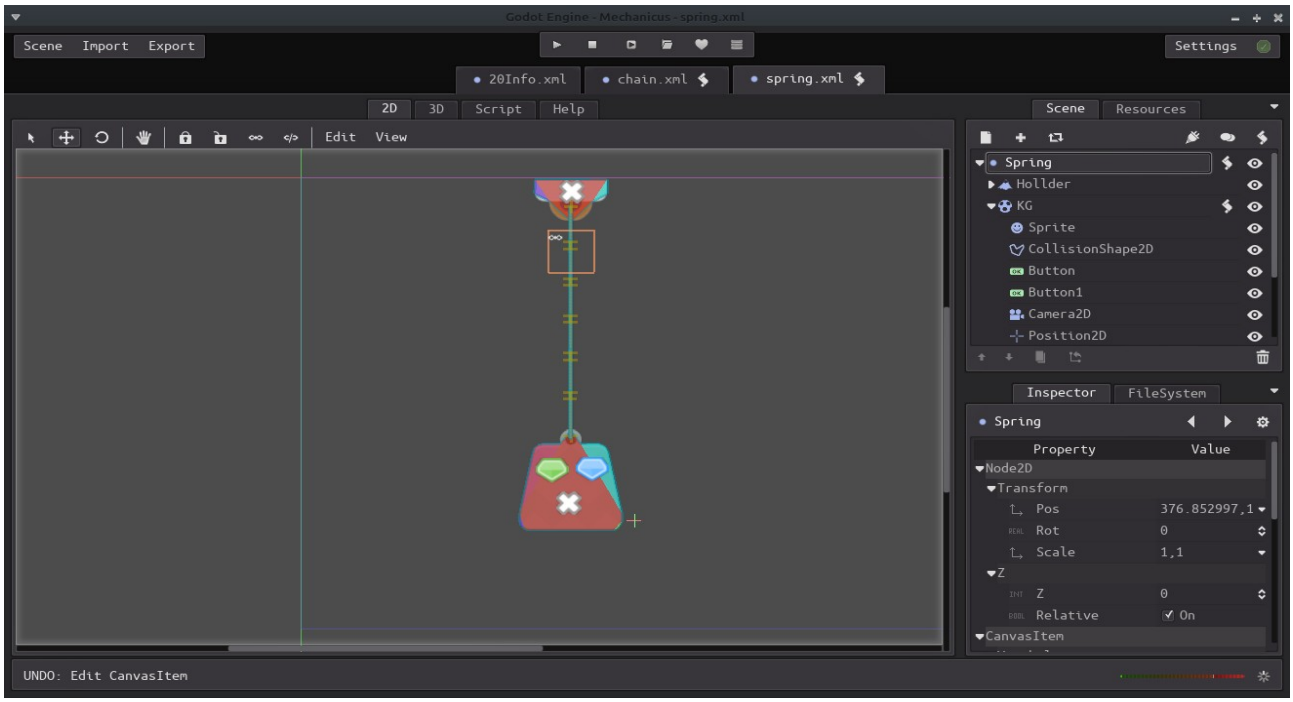


Rys. 29: Plik chain.xml w edytorze

Scena odwzorowująca wahadło składa się z miejsca mocowania, łańcucha oraz kuli. Poszczególne pary elementów połączone są obiektem PinJonit2D. Jak było już wcześniej wspominanie w rozdziale **5.1.3** oprócz omówionej przed chwilą wersji posiada jeszcze wersję o uproszczonej fizyce, która jest wykorzystana podczas ustalania pozycji tej maszyny na planszy. Zabieg taki został podjęty przez autora z powodu, iż bardzo trudno byłoby zaimplementowanie jednoczesnego i jednostajnego ruchu wszystkich elementów wahadła a dodatkowo byłoby to sporym obciążeniem dla komputera.

5.2.4 Masa na sprężynie

W przypadku masy na sprężynie, podobnie jak w przypadku próby zaimplementowania wahadła matematycznego okazało się, iż o wiele prościej jest odwzorować odważnik na sprężystej line niż na sprężynie. Wynika to z specyfiki obiektu DampedSpringJonit2D, który odwzorowuje sprężynę łączącą dwa wybrane obiekty. Za jego pomocą o wiele łatwiej było stworzyć sprężystą linę, niż mechanikę pojedynczej sprężyny.



Rys. 30: Plik spring.xml

Taka lina działa jak połączenie wahadła fizycznego ze sprężyną. Tutaj oprócz PinJoint2D (łączenia: mocowanie – lina, lina – odważnik), jest wykorzystywany DampedSpringJonit2D.

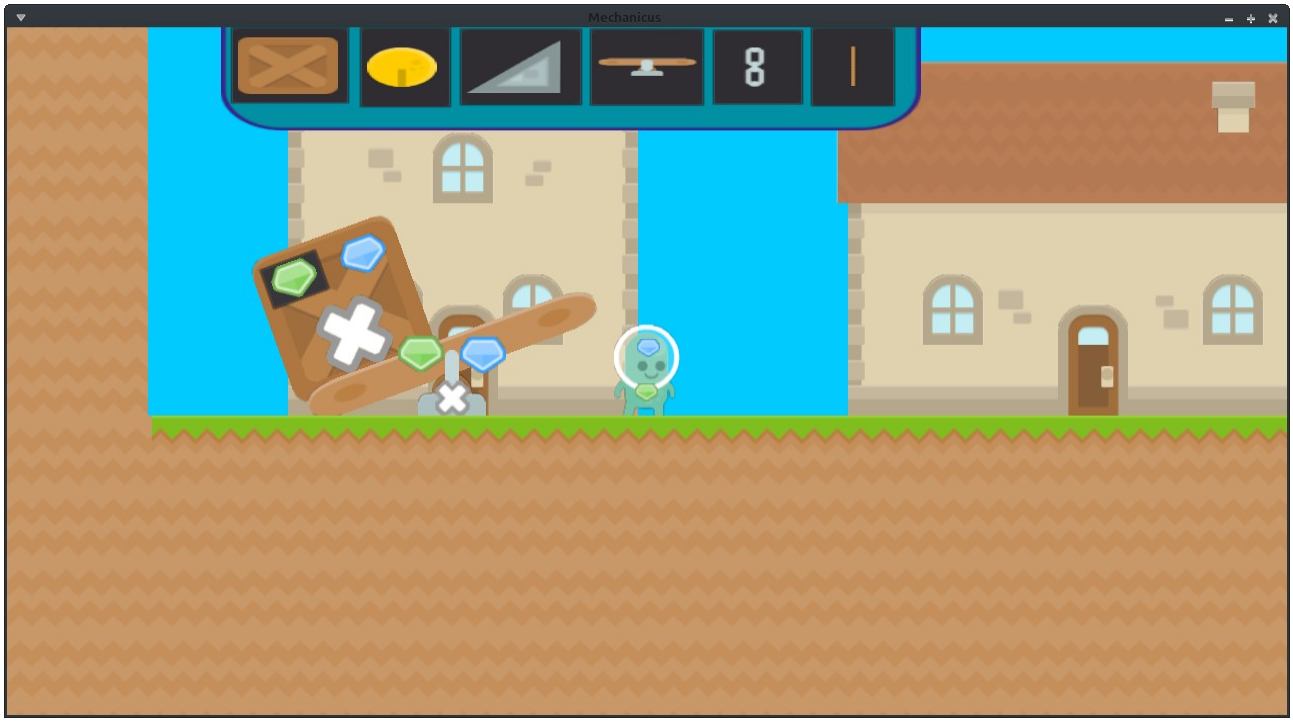
W przypadku tego modelu też jest wykorzystywana uproszczona wersja podczas ustalania położenia.

5.3 Grafika i dźwięk

Rys. 31: Strona http://www.kenney.nl w przeglądarce Firefox

Autor skorzystał z gotowej grafiki dostępnej na licencji Creative Commons i dostępnej na stronie http://www.kenney.nl. Użytą bitmapową czcionkę wygenerowano na podstawie dostępnej na wolnej licencji czcionki Ubuntu font.ubuntu.com. Autor uświadomił sobie również w trakcie realizacji projektu, iż nie zdoła stworzyć tak mocno rozbudowanej gry jak to pierwotnie planował i postanowił zrezygnować z dźwięku w projekcie.

6 Zakończenie



Rys. 32: Aplikacji Mechanicus w działaniu

Założeniem projektu było stworzenie edukacyjnej gry komputerowej, która przede wszystkim pokaże oferowane przez silnik Godot możliwości w zakresie odwzorowania zjawisk fizycznych. Wybrany sposób realizacji doprowadził do powstania w pełni funkcjonalnego prototypu gry z maszynami, z którymi można wchodzić w interakcję.. Gra tworzona była od najprostszego modelu maszyny do bardziej złożonych. Na każdym kolejnym etapie był dodawane nowe funkcje. Po każdym kroku starano się jak najlepiej przetestować wprowadzone zmiany. Jednak z powodu ograniczonego czasu i wysokiego stopnia rozbudowania projektu pewne błędy, nie zmniejszające jej funkcjonalności, pojawiają się w grze.

Ze względu na małą unikatowość użytego rozwiązania, a więc brak literatury dotyczącej tego jak zaimplementować pewne elementy autor często otrzymywał maszyny o wiele bardziej złożone niż wynikało to z założeń projektu. W trakcie realizacji autor nauczył się i przekonał, że część elementów w grze można by było zrealizować szybciej i prościej zachowując jednocześnie wysoki stopień skomplikowania maszyn i odwzorowania zjawisk fizycznych niż początkowo zakładał. Autor też wiele razy musiał weryfikować swoje założenia i pisać pewne skrypty od nowa, dlatego też cześć z nich w nazwach posiada numer lub jakiś sufiks.

Realizacja projektu gry *Mechanicus* wiązała się z koniecznością wirtualnego odwzorowania maszyn mechanicznych oraz zaprojektowania i zaprogramowania interfejsu pozwalającego na przeprowadzanie eksperymentów z ich użyciem. Wymagało to wykorzystania znacznej części wiedzy i umiejętności pozyskanych w toku studiów. Mając znikome doświadczenie w tworzeniu gier komputerowych, konieczne okazało się również dokształcenie w tej dziedzinie. Praca nad projektem trwała kilka miesięcy i przekładała się na wiele godzin spędzonych przed komputerem. Własnoręcznie napisane skrypty składają się ostatecznie z ponad 26 tysięcy znaków, pomijając wygenerowane przez Godot pliki xml zawierające opis struktury scen. Jak na jedną z pierwszych gier komputerowych, projekt okazał się być dużym wyzwaniem i prawdopodobnie, gdyby nie intuicyjność Godota, nigdy nie zostałby ukończony. Mimo wszystko okazało się, iż taki rodzaj pracy daje zaskakującą satysfakcję.

6.1 Przyszłość projektu

Mechanicus składa się obecnie jedynie poziomu pozwalającego przetestować zaimplementowane maszyny. Dzięki temu sprawdzono, że Godot jest dostatecznie dobrym narzędziem do stworzenia gry o charakterze edukacyjnym. Jeśli projekt rozwijałby się dalej to należałoby uprościć jego strukturę, udźwiękowić, stworzyć poziomy z zagadkami do których rozwiązania trzeba byłoby wykorzystać dostępne maszyny i dodać wciągającą fabułę podobnie jak miało to miejsce we wspominanej grze *Physicus.* Dodatkowo można byłoby zaprogramowywać edytor poziomów i maszyn, aby dać możliwość użytkownikom tworzenia ciekawych eksperymentów w grze. Inną opcją na rozwój gry jest opublikowanie jej kodu źródłowego na portalu Github.

7 Bibliografia

1. „Grywalizacja jak zastosować mechanizmy gier w działaniach marketingowych”, Paweł Tkaczyk, Wydawnictwo HELION 2012, Gliwice
2. „Wprowadzenie do dydaktyki ogólnej”, Wincenty Okoń, Wydawnictwo akademickie Żak 1998, Warszawa
3. „Modelowanie i symulacja komputerowa w mechatronice : przykłady”, Tomasz Kiczkowiak, Tomasz Tarnowski, Wojciech Tarnowski, Zenon Ociepa, Wydawnictwo Politechnik Koszalińskiej 2009, Koszalin
4. „Mechanika ogólna tom 1 Statyka i kinematyka”, Jan Misiak, Wydawnictwo Naukowo Techniczne 1998, Warszawa
5. „Ciekawi świata 1 Fizyka Podręcznik zakres rozszerzony”, Grzegorz Kornaś, Wydawnictwo Pedagogiczne OPERON 2012, Gdynia
6. https://pl.wikipedia.org/Silnik_gry – data dostępu: 22.07.2015
7. https://en.wikipedia.org/wiki/Unity_(game_engine)– data dostępu: 17.07.2015
8. https://en.wikipedia.org/wiki/Unreal_Engine – data dostępu: 17.07.2015
9. https://en.wikipedia.org/wiki/Godot_(game_engine)– data dostępu: 17.07.2015
10. http://www.nyu.edu/about/news-publications/publications/connect-information-technology/2012/04/30/video-games-and-the-future-of-learning.html - data dostępu: 17.07.2015
11. http://polska.newsweek.pl/wyniki-egzaminu-gimnazjalnego-2015-cke-publikuje-wyniki,artykuly,365345,1.html - data dostępu: 17.07.2015
12. http://www.fakt.pl/wydarzenia/matura-2015-wyniki-matury-2015,artykuly,555707.html - data dostępu: 17.07.2015
13. http://www.perspektywy.pl/index.php?option=com_content&task=view&id=4049&Itemid=106 - data dostępu: 17.07.2015

14. http://www.edunews.pl/nowoczesna-edukacja/edutainment/2111-rola-gier-komputerowych-i-edutainment-w-edukacji - data dostępu: 12.07.2015

15. http://www.przygodoskop.pl/417/encyklopedia.htm - data dostępu: 12.07.2015

16. http://gry.onet.pl/zapowiedzi/physikus-return/p53zk - data dostępu: 12.07.2015

17. https://pl.wikipedia.org/wiki/Równia_pochyła - data dostępu: 22.07.2015

18. https://pl.wikipedia.org/wiki/Dźwignia - data dostępu: 22.07.2015

19. https://pl.wikipedia.org/wiki/Wahadło - data dostępu: 22.07.2015

20. https://pl.wikipedia.org/wiki/Krążek_stały - data dostępu: 22.07.2015

21. https://pl.wikipedia.org/wiki/Prędkość - data dostępu: 22.07.2015

22. https://pl.wikipedia.org/wiki/Prędkość_kątowa - data dostępu: 22.07.2015

23. http://www.godotengine.org/ - data dostępu: 15.07.2015

24. https://github.com/okamstudio/godot/wiki - data dostępu: 22.07.2015

25. https://github.com/okamstudio/godot/graphs/contributors - data dostępu: 22.07.2015

26. http://www.newmedia.org/game-based-learning--what-it-is-why-it-works-andwhere-its-going.html - data dostępu: 22.07.2015

27. http://www.kenney.nl - data dostępu: 22.06.2015

28. https://pl.wikibooks.org/wiki/Fizyka_dla_liceum/Dynamika_bryły_sztywnej – data dostępu: 3.08.2015

29. https://tools.wmflabs.org/blogconverter/ – data dostępu: 3.08.2015

30. http://www.garyshood.com/htmltobb/ – data dostępu: 3.08.2015

31. http://font.ubuntu.com – data dostępu: 3.08.2015