LABORATORIUM ZTPGK – FIZYKA (MINI PROJEKT)

Maciej Kowalik

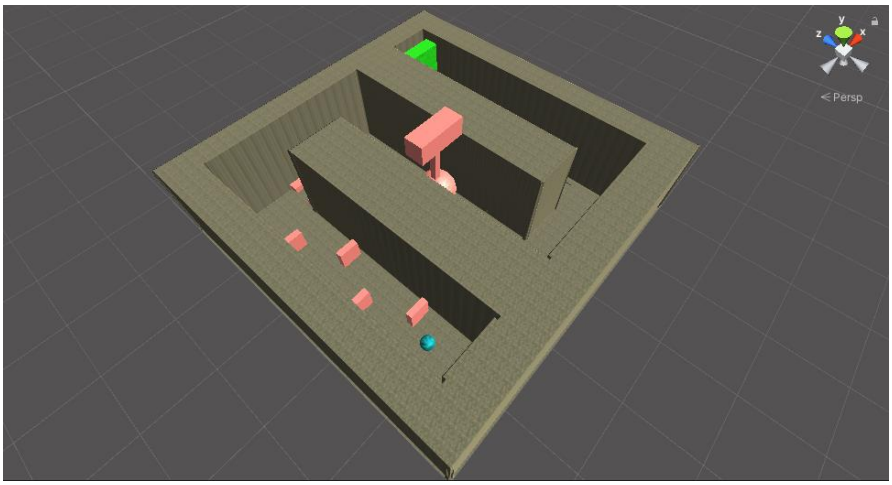
Tomasz Kaczmarzyk

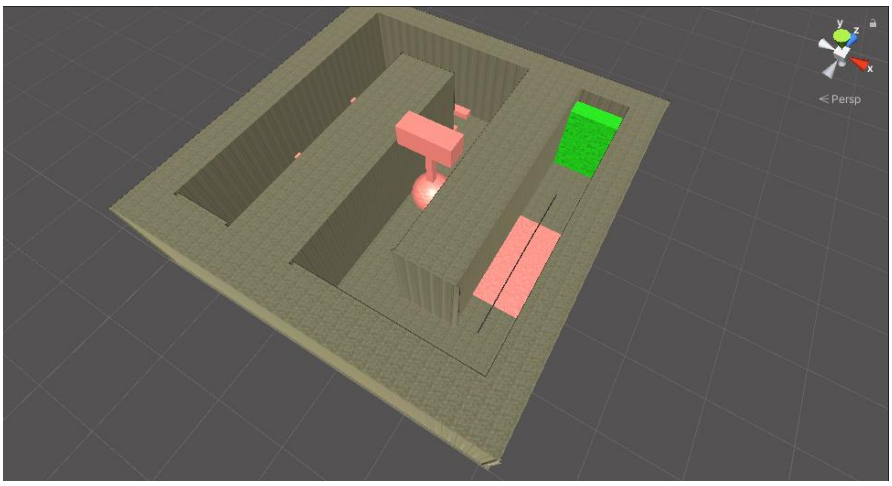
Opis działania gry

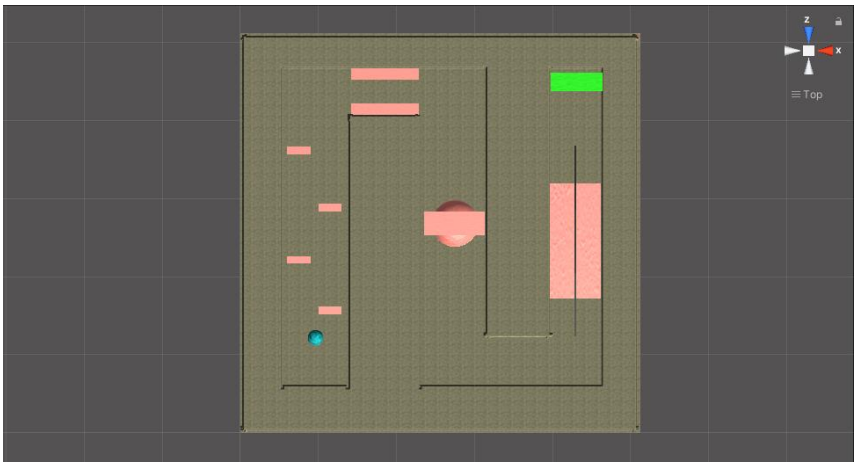
Zaprojektowana gra polega na pokonaniu toru przeszkód i dotarciu do mety. Gracz steruje kulką i musi unikać przeszkód czyhających na gracza w terenie oraz pokonywać trudności toru. Na przykładowej mapie zaprojektowano ruchome bloki, wahadło jak i most do pokonania. W przypadku zetknięcia się z przeszkodą lub spadnięciu z mostu gra kończy się niepowodzeniem, a w przypadku dotarcia do mety, gracz wygrywa.

Specyfikacja zewnętrzna

Mapa, na której porusza się gracz ma kształt labiryntu. W kolejnych korytarzach umieszczone są różne przeszkody, mające kolor czerwony. Kolorem zielonym oznaczony jest blok, do którego należy dotrzeć, aby ukończyć grę.







Pierwszą przeszkodą są ruchome bloki, z którymi gracz nie może się zetknąć. Następnie znajdują się bariery, między którymi gracz musi się przedostać. Kolejna przeszkoda to wahadło, którego należy uniknąć, a ostatnią przeszkodą jest do pokonania most. Powierzchnia pod mostem imituję przepaść, z którą gracz nie może się zetknąć. Ostatni element to obiekt oznaczający dotarcie do mety.

Gracz porusza kulką za pomocą strzałek lub klawiszy WSAD oraz klawisza spacja do wykonania skoku.

W przypadku niepowodzenia, gdy gracz nie pokona przeszkody, gra kończy się a na ekranie otrzymujemy następujący komunikat o porażce wraz z przyciskiem do wyłączenia gry.

W przypadku pokonania toru i dotarcia do mety, gra kończy się a na ekranie otrzymujemy następujący komunikat o wygranej wraz z przyciskiem do wyłączenia gry.

Specyfikacja wewnętrzna

Stworzenie terenu do gry zostało oparte o wcześniej utworzony w ramach laboratorium z terenu skrypt, generujący teren na podstawie pliku .jpg. Skrypt *TerrainScript.cs* pobiera informacje o pikselach na obrazie i jeżeli dany piksel nie jest on biały to podnosi dany fragment terenu. To rozwiązanie pozwala na sprawne generowanie wielu, różnych terenów w przypadku rozszerzania gry o kolejne poziomy.

Sterowanie kulką zapewnione jest w skrypcie *BallMovement.cs*. Poruszanie się gracza zostało zaimplementowane za pomocą kontrolera *CharacterController*, metody *move*. Ustawione zostały wartości prędkości gracza, grawitacji jak i szybkości skoku. Za graczem cały czas podąża kamera z perspektywy trzeciej osoby. W skrypcie *CameraController.cs* zaimplementowane jest stałe powiązanie z obiektem, za którym kamera podąża oraz ważnym aspektem jest zsynchronizowanie rotacji obiektu i kamery, aby kamera zawsze była skierowana w kierunku ruchu kulki. Dodatkowo możemy jako parametry ustawić offset kamery, jak daleko ma być oddalona od obiektu.

Ruch pierwszej przeszkody, będącej poruszającymi się blokami, jest zaimplementowany w skrypcie *CubesMovement.cs*. W skrypcie określone są pozycje początkowe oraz końcowe obiektów, a ich poruszanie zapewnione jest za pomocą metody *addForce*, w danej osi. Jest to zastosowanie siły, czyli jednego z elementów silnika fizycznego wskazanego w instrukcji laboratoryjnej. Obiektom nadawana jest siła w danym kierunku i o odpowiednim znaku, czego efektem jest rozpędzanie się przeszkody. Gdy osiągnie ona pozycję końcowa, znak siły odwraca się i rozpędza się i wraca w miejsce początkowe.

Ruch przeszkody będącej wahadłem jest zaimplementowany w skrypcie *PendulumMovement.cs*. Samo wahadło składa się z trzech części: górnego obrotowego trzonu, wahadła oraz kuli umieszczonej na jego zakończeniu. Połączenie obiektów zostało zapewnione za pomocą kolejnego z elementów silnika fizycznego wskazanego w instrukcji czyli *Joints*. Elementy zostały ze sobą złączone

za pomocą komponentu *FixedJoint*, dzięki czemu są one sztywno połączone ze sobą, a rotacja górnego trzonu pozwala na ruch wahadłowy złączonym obiektom.

Kolejny element silnika fizycznego, który został zastosowany w projekcie to *Triggery*. Za ich pomocą sprawdzane jest czy gracz nie zetknął się z przeszkodą lub obiektem oznaczającym metę. Zachowanie pod zderzeniu z przeszkodą jest zaimplementowany w skrypcie *ObstaclesTouch.cs*, a z metą w *FinishScript.cs*. Do sprawdzenia użyta została metoda *OnTriggerEnter*, gdzie sprawdzane jest czy obiekt, który spowodował wywołanie funkcji to gracz. Dla przeszkód w komponencie *BoxColider* należało zaznaczyć opcję *isTrigger*. W takim przypadku gra zatrzymuje się, a na ekranie pokazuje się odpowiedni komunikat, w zależności od wyniku gry – flaga odpowiedzialna za widoczność odpowiedniego GUI jest zmieniana.

Testowanie

Gra działa zgodnie z oczekiwaniami i wszelki kontakt z przeszkodą czy obiektem mety kończy rozgrywkę. Gra była testowana pod kątem możliwości jej przejścia, należało dostosować zarówno prędkość gracza jak i ruchomych przeszkód, aby możliwym było pokonanie całego toru przeszkód.

Zanim gra została doprowadzona do stanu końcowego, wykonano wiele różnych testów związanych z poruszaniem obiektem gracza. Zanim zdecydowano się na wykorzystanie *CharacterController*, sterowanie graczem było zapewnione za pomocą edytowania własności *RigidBody*. Jednak po zaznajomieniu się z wyżej wspomnianym kontrolerem i sprawnym sposobem zaimplementowania poruszania jak i podłączenia do niego kamery, zrezygnowano z *RigidBody*. Po czasie przy dodawaniu przeszkód na torze, zauważono, że dwa wskazane rozwiązania realizacji obiektu gracza są bardzo odmienne w przypadku interakcji z innymi obiektami.

Podsumowanie

Tworzony projekt znacznie poszerzył wiedzę w zakresie tworzenia obiektów poruszających się jak i ich wzajemnych interakcji. Można było zapoznać się z działaniem wielu elementów silnika fizycznego oraz zaimplementować je do gry. Do realizacji prostej gry, warto użyć komponentu *CharacterController* do sterowania postacią, jest ono zdecydowanie prostsze do implementacji. Jednakże, jeżeli projekt miałby być rozwijany, lepszym wyjściem byłoby skorzystanie z *RigidBody* ponieważ dużo zapewnia on dużo więcej funkcji do interakcji z fizyką gry. Dodatkowo generowanie terenu z plików .jpg jest również wygodnym rozwiązaniem, ponieważ budowa prostych poziomów wymaga jedynie utworzenia obrazu z układem ścian.

Link do repozytorium

https://github.com/macikow202/ZTPGKproj