Adam Borowski, nr albumu 137252
Radek Pieczątkiewicz, nr albumu 137373

Msiball

gra Unity wykorzystująca moduł rozpoznawania emocji

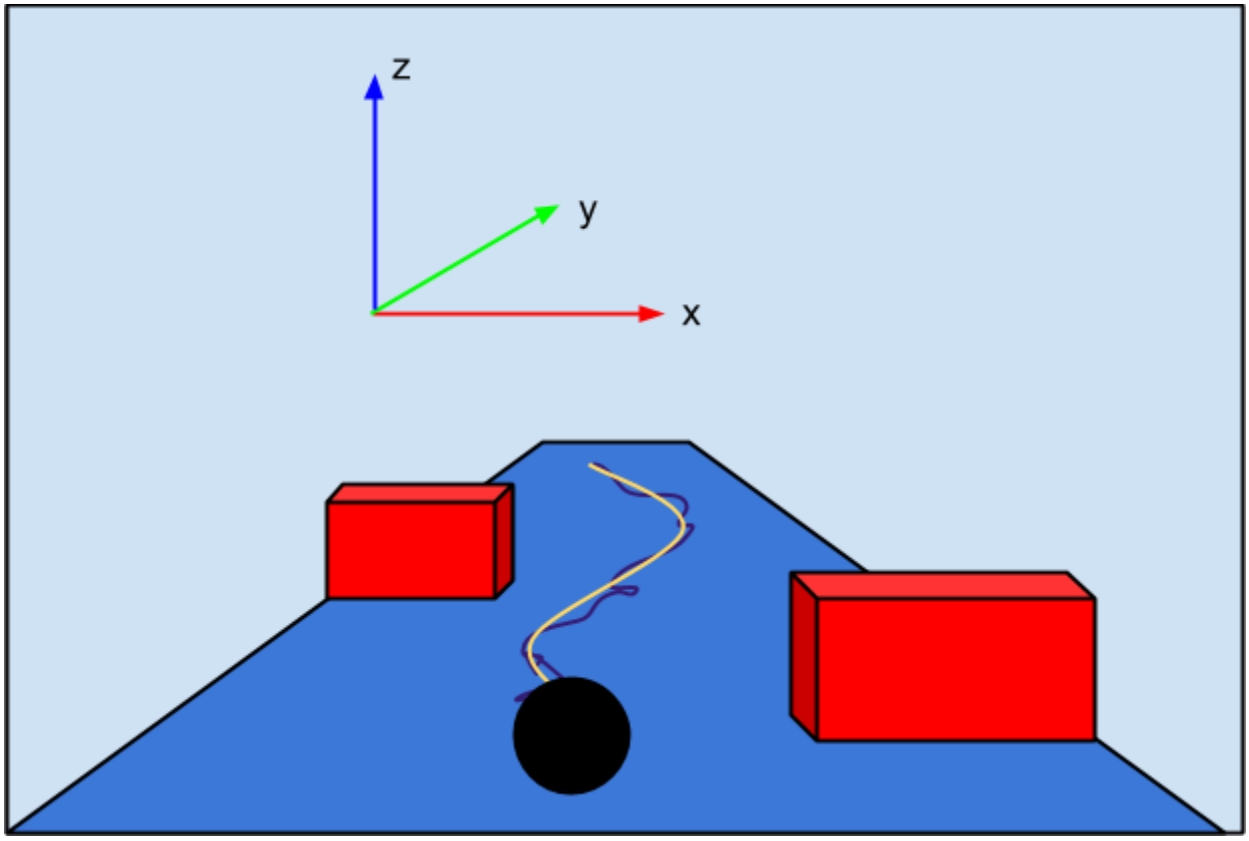
Analiza wymagań

Opis wymagań

Celem aplikacji jest demonstracja wykorzystania silnika rozpoznawania emocji ER dostarczonego przez Katedrę Inteligentnych Systemów Interaktywnych.
PowerMSI jest trówymiarową grą zręcznościową, w której poziom trudności regulowany jest przez szereg czynników przetwarzanych przez zewnętrzny moduł ER.
Gracz steruje kulą w dwóch kierunkach: prawym i lewym. Kula toczy się po równi pochyłej zwiększając swoją prędkość. Na trasie gracz otrzymuje punkty za zbieranie żetonów oraz za kolejne przebywane odcinki trasy. Na trasie występują liczne przeszkody, które użytkownik musi unikać. W przypadku kolizji użytkownik traci pewną ilość punktów.

Ważniejsze cechy gry

- trasa gry jest generowana dynamicznie w zależności od emocji gracza
- gracz widzi scenę trójwymiarową w ten sposób, że przemieszcza się w głąb ekranu, pozycja kuli na ekranie jest stała w pionie i zmieniana w poziomie przez użytkownika



- Gra toczy się w nieskończoność a celem gry jest pobicie kolejnych rekordów punktowych podczas bezkolizyjnego pokonywania trasy
- Gra udostępnia dwa tryby rozgrywki:
    - z uwzględnieniem emocji podawanych przez ER (ułatwianie graczowi początkującemu i podnoszenie poziomu trudności graczowi zaawansowanemu)
    - bez uwzględnienia modułu ER, ale z bonusami które gracz może zdobyć w celu ułatwienia sobie gry

Jakie parametry środowiska gry są przekazywane do ER

- częstość kolizji z przeszkodami
- chaotyczność sterowania (średnie odchylenie realnej (kolor fioletowy na rysunku) trasy od aproksymowanej (kolor żółty))
- częstotliwość naprzemiennych uderzeń w klawisze kierunków
- stosunek liczby zebranych żetonów do liczby ominiętych żetonów

Na jakie aspekty gry mają wpływ emocje gracza rozpoznawane przez ER

- kąt równi pochyłej
- bezwładność sterowania (ociężałość kuli)
- rozmiar kuli
- czestość pojawiania się przeszkód
- wielkość przeszkód
- ruchomość przeszkód

Dodatkowe wymagania

Gra powinna umożliwiać realizację sceny 3D w czasie rzeczywistym. Aplikacja powinna być napisana zgodnie z aktualnie panującymi paradygmatami projektowania obiektowego i wzorcami projektowymi.
Kod aplikacji powinien być samodokumentujący się.
Produkt ma być opublikowany na licencji Open Source tak aby był zgodny z licencją modułu ER dostarczonego przez KISI. Kod aplikacji Unity powinien być umieszczony na portalu GitHub.

Gra powinna wykorzystać conajmniej jeden z dwóch trybów działania modułu ER: wektor emocji oraz model psychologiczny PAD (Pleasure, Aurosal and Dominance)

Docelowa grupa użytkowników

Gra dedykowana jest użytkownikom casualowym i zaawansowanym gdyż jest prosta w obsłudze (two-button-game) oraz dostosowuje poziom trudności w zależności od emocji gracza. Opcje udostępnione przez aplikacją pozwolą dostoswać poziom trudności tak, aby gra była grywalna dla każdego gracza.

Adam Borowski, nr albumu 137252
Radek Pieczątkiewicz, nr albumu 137373

Platforma sprzętowa

Gra może być uruchomiona na komputerze osobistym (PC) z systemem operacyjnym Windows. Dodatkowym wymaganiem może być Microsoft Kinect, jeśli moduł ER dostarczony przez KISI wspiera takie rozwiązanie.

Środowisko pracy

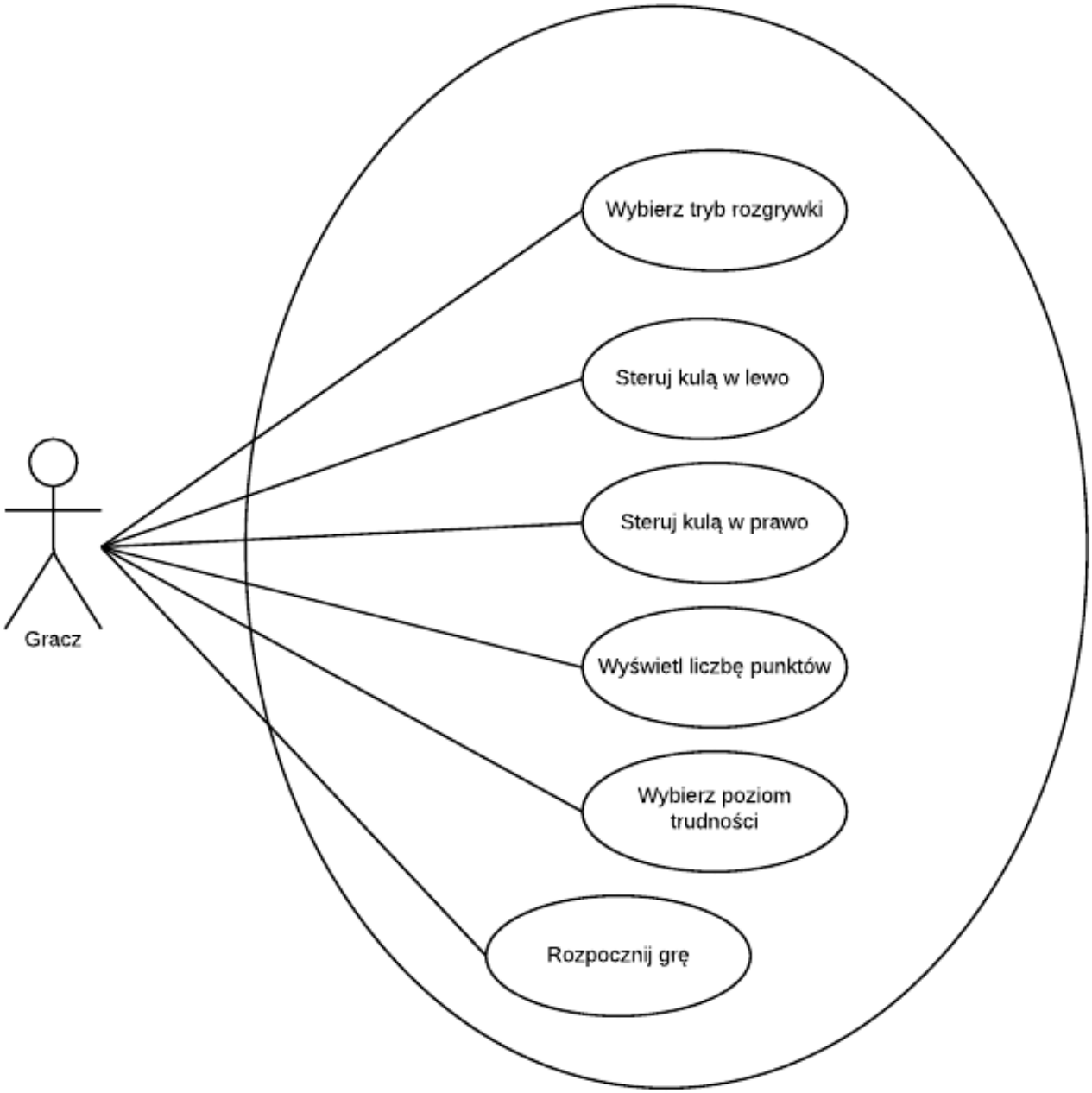
W ramach projektu zostanie wykonana gra w Unity SDK. Moduł ER dostarczony przez KISI będzie rozwijany w środowisku Microsoft Visual Studio. Wykorzystane protokoły zależą od specyfikacji modułu ER dostarczonego przez KISI.

Adam Borowski, nr albumu 137252
Radek Pieczątkiewicz, nr albumu 137373

Projekt systemu

Architektura

Diagram przypadków użycia



PU-1	Wybierz tryb rozgrywki
stan gry	otwarty ekran powitalny
przebieg	użytkownik wybiera jedną z dwóch opcji poprzez kliknięcie na jendym z dwóch przycisków: ● gra ze wsparciem rozpoznawania emocji ● gra bez wsparcia rozpoznawania emocji
wynik	zmieniony zostanie tryb rozgrywki, przycisk odpowiadający wybranemu trybowi zostanie wyróżniony, ekran powitalny pozostanie aktywny

PU-2	Wybierz poziom trudności
stan gry	otwarty ekran powitalny, wybrany tryb **bez wsparcia ERF**
przebieg	użytkownik wybiera jedną z dwóch opcji poprzez kliknięcie na jendym z dwóch przycisków: ● poziom trudny ● poziom łatwy
wynik	zmieniony zostanie poziom trudności, przycisk odpowiadający wybranemu poziomowi trudności zostanie wyróżniony, ekran powitalny pozostanie aktywny

PU-3	Rozpocznij grę
stan gry	otwarty ekran powitalny
przebieg	gracz wybiera przycisk “Graj” który powoduje przejście do modułu GameScreen
wynik	ekran powitalny zostanie usunięty, ekran gry zostanie załadowany

PU-4	Steruj kulą w lewo
stan gry	otwarty ekran gry, gracz naciska na klawiaturze przycisk strzałki w lewo
przebieg	przyśpieszenie boczne kuli zostanie zmienione
wynik	kula z pewną bezwładnością zaczyna zmierzać do lewej częsci toru

PU-5	Steruj kulą w prawo
stan gry	otwarty ekran gry, gracz naciska na klawiaturze przycisk strzałki w prawo

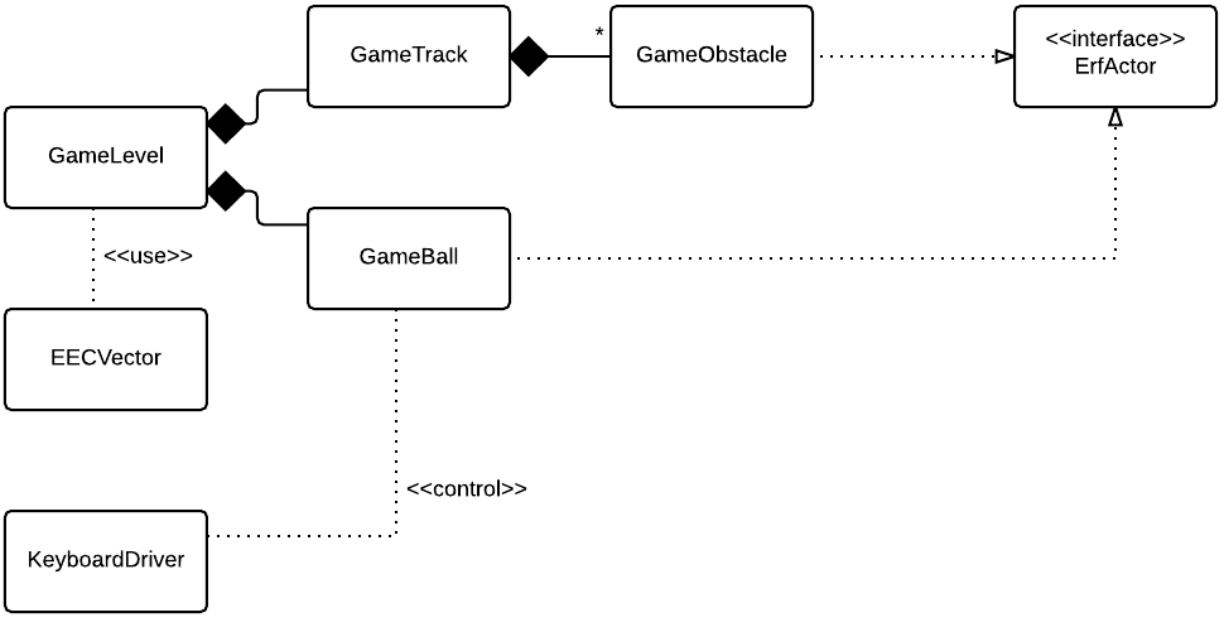
przebieg	przyśpieszenie boczne kuli zostanie zmienione
wynik	kula z pewną bezwładnością zaczyna zmierzać do prawej częsci toru

PU-6	Wyświetl liczbę punktów
stan gry	otwarty ekran gry
przebieg	użytkownik znajduje panel z liczbą punktów w rogu ekranu
wynik	b/z

W systemie zaprojektowano 2 moduły

1. WelcomeScreen - ekran powitalny 2D przygotowujący rozgrywkę
2. GameScreen - ekran gry 3D realizujący główne funkcje systemu

Diagram klas modułu GameScreen



Interfejs	ErfActor
opis	Interfejs reprezentujący Aktora który jest połączony z ERF

Klasa	GameLevel
opis	Klasa kontrolująca obiekt piłki i trasy, pełni rolę mediatora i inicjalizatora

Msiball

Klasa	EECVector
opis	Klasa wektora danych emocji przekazywanych z/do ERF

Klasa	KeyboardDriver
opis	Klasa odbierająca zdarzenia klawiszy od użytkownika. Przekształca te zdarzenia na modyfikację parametrów przyśpieszenia normalnego.

Klasa	GameTrack
opis	Klasa kontrolująca tworzenie, usuwanie i rozmieszczanie przeszkód na powierzchni, odpowiada za uzyskanie wrażenia nieskończoności trasy.

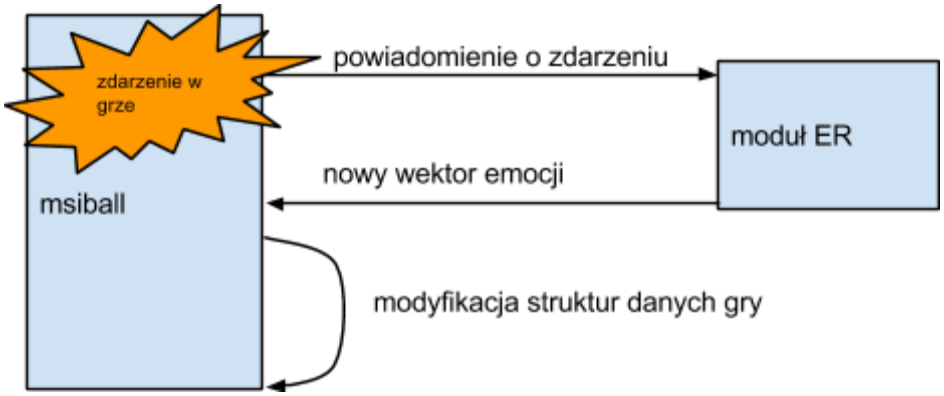
Klasa	GameObstacle
opis	Klasa reprezentująca przeszkodę na trasie. Implementuje interfejs Aktora, gdyż przeszkoda swoje zachowanie uzależnia od wektora emocji pochodzącego z ERF

Klasa	GameBall
opis	Klasa reprezentująca kulę sterowaną przez gracza. Implementuje interfejs Aktora, gdyż kula uzależnia wpływ gracza od wektora emocji pochodzącego z ERF

Uwzględnienie modułu ERF

a. zewnętrzny moduł ERF

Komunikacja została pokazana na poniższym diagramie:



b. zewnętrzny moduł szacujący emocje

Jest to biblioteka DLL napisana w języku C++, która będzie ładowana do gry poprzez:

- załadowanie biblioteki z komponentami do kontekstu Erf
- ErfContext.GetInstance ().LoadComponentLibrary ("MSI_SZACUNEK.dll");
- pobranie eksperta: ErfContext.GetInstance().FindExpert(“MSI_SZACUNEK”),
- kontekst sam zadba o zarządzanie załadowanymi dodatkami

c. informacje o zdarzeniach wpływające na emocje

```
enum GameEvent {
        STD_PLAYER_EMOTIONAL_STATE_CHANGE, ///stan emocji gracza został zmieniony
        STD_TIME_ELAPSED, // dokonano pomiaru danych z GameState oraz KeyboardDriverState
        STD_LEVEL_LOADED, //rozpoczęto grę od nowa
        MSI_COLLISION, //zderzenie kuli z przeszkodą
       STD_OBSTACLES_CLOSE, // zagrożenie zderzeniem kuli z przeszkodą
        EXT_GAME_EVENT //koniec rozgrywki
};
```

Adam Borowski, nr albumu 137252
Radek Pieczątkiewicz, nr albumu 137373

Środowisko wytwarzania

- System operacyjny: Windows 8.1

Rozwijanie gry Unity

- język programowania: CSharp
- SDK + IDE: Unity 4.3.2f1, MonoDevelop 4.0.1
- wyjściowy szablon projektowy: gra ErfDemonstration załączona do ERF

Rozwijanie ERF

- język programowania: C++
- SDK + IDE: Visual Studio 2010 Professional
- wyściowy szablon projektowy: projekt SLN frameworka ERF

Stuktury danych

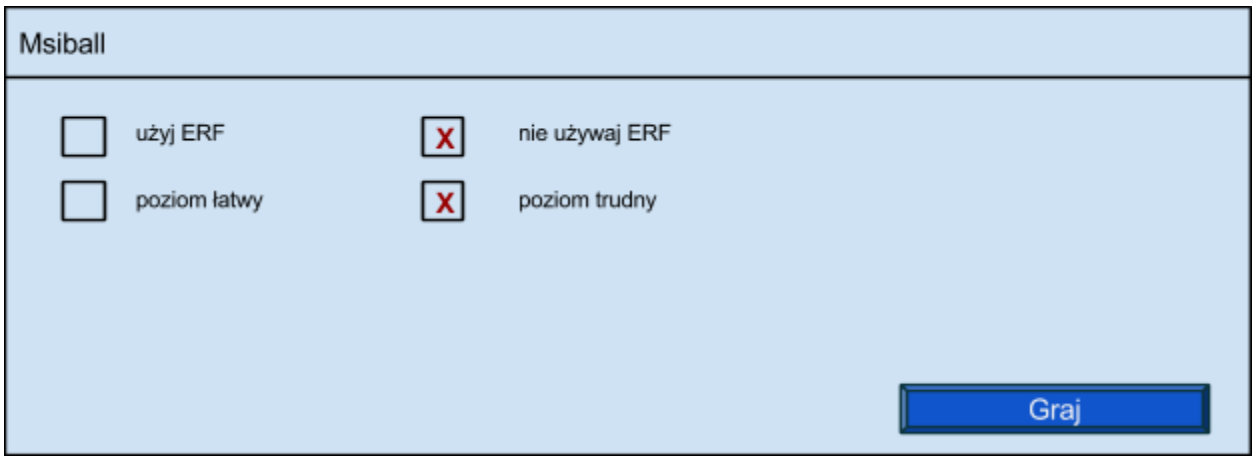
nazwa	KeyboardDriverState
opis	struktura opisująca stan klawiszy w momencie t
pola	• LeftHoldDuration - czas w milisekundach mówiący od jak dawna gracz ma wciśnięty klawisz LEFT • RightHoldDuration - czas w milisekundach mówiący od jak dawna gracz ma wciśnięty klawisz RIGHT • LeftPressCount - ilość zdarzeń wciśnięcia klawisza LEFT na odcinku czasu • RightPressCount - ilość zdarzeń wciśnięcia klawisza RIGHT na odcinku czasu • PressFrequencyMeasTime - czas mierzenia parametrów leftPressCount oray rightPressCount

nazwa	GameState
opis	struktura opisująca stan osiągnięć gracza w momencie t
pola	• PointsCount - licznik punktów uzyskanych przez gracza • TimeCount - aktualny czas rozgrywki • Level - obecny poziom • LastEmotionVector - ostatni wektor emocji przypisany aktorowi kuli • LastMoodVector - ostatni wektor nastroju przypisany aktorowi kuli

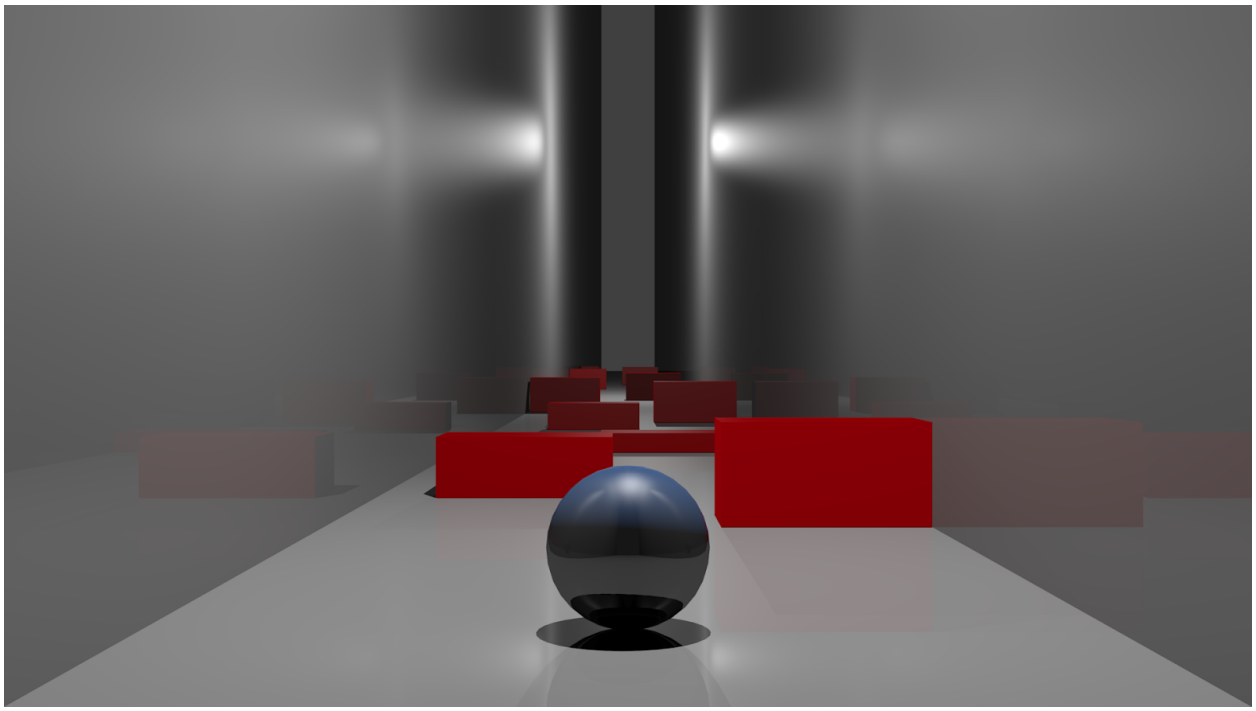
Adam Borowski, nr albumu 137252
Radek Pieczątkiewicz, nr albumu 137373

Interfejs użytkownika

WelcomeScreen



GameScreen



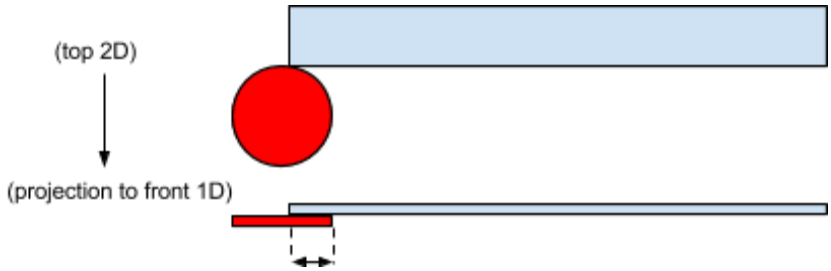
Adam Borowski, nr albumu 137252
Radek Pieczątkiewicz, nr albumu 137373

Wykorzystywane zasoby

Wykorzystany zostanie framework ERF wraz z zależnymi bibliotekami boost i SWIG

Algorytm wyznaczania stopnia powagi kolizji ProjectionCoverage

Celem tego algorytmu jest rozpoznanie, do jak wielkiej kolizji doszło z przeszkodą. Istotne bowiem jest, czy kula tylko “otarła się” czy dosłownie “wbiła się w ścianę”. Aby wyznaczyć ten współczynnik należy dokonać rzutowania obiektów wpierw na płaszczyznę xz, następnie rzutować na oś x.



W wyniku podwójnej projekcji otrzymamy dwa odcinki na osi. Wynikiem algorytmu jest stosunek długości cześci wspólnej projekcji kuli i ściany do długości projekcji kuli.

Opis głównych funkcji systemu (wywołania i parametry)

Funkcja	GenerateObstacle
opis	generowanie kolejnej przeszkody w grze
parametry	*Przedział wartości parametrów jest skalowany do przedziału o granicach wyznaczonych przez wartości minimalne i maksymalne zdefiniowane w kodzie programu w postaci stałych.* ● x1 - odległość lewej ściany przeszkody od lewej ściany toru <0,1> ● x2 - odległość prawej ściany przeszkody od lewej ściany toru <0,1> ● z - wysokość przeszkody <0,1> ● dy - odległość od poprzedniej przeszkody

Funkcja	RegisterCollision
opis	reakcja na zderzenie kuli z przeszkodą w wyniku której powinno zostać zgłoszone zdarzenie EEC
parametry	● CurrentSpeed - prędkość zderzenie z przeszkodą ● ProjectionCoverage - część wspólna rzutów kuli i przeszkody (0,1> patrz algorym ProjectionCoverage ● SuccesfulyAvoidedObscatles - ile było uprzednio ominiętych przeszkód

	● KeyboardDriverState - stan wciśnięcia klawiszy gracza (na tej podstawie jesteśmy w stanie dowiedzieć się, czy gracz próbował ominąć przeszkodę)

Funkcja	ReceiveEECVector
opis	Obsługa wektora emocji i nastroju, przekazuje wektory do poszczególnych obiektów w celu zmiany parametrów rozgrywki
parametry	● EmotionVector ● MoodVector

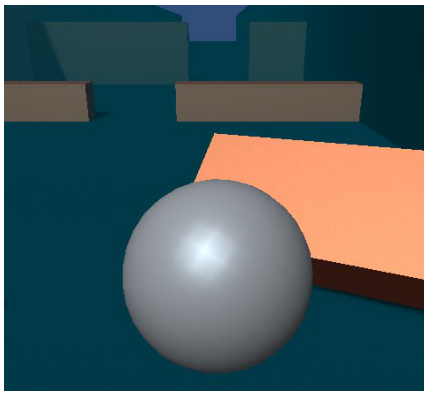
Funkcja	ProcessScene
opis	Modyfikacja położenia kuli oraz toru na podstawie aktualnych wartości rozgrywki i zmiennych sterujących.
parametry	● XPosition - aktualna pozycja kuli na osi x ● XSpeed - aktualna prędkość kuli na osi x ● XAcceleration - aktualne przyspieszenie kuli na osi x ● XAccelerationDelta - modyfikator przyspieszenia zależny od użytkownika i poziomu trudności ● YPosition - aktualna pozycja kuli na osi y ● YSpeed - aktualna prędkość kuli na osi y ● YAcceleration - aktualne przyspieszenie kuli na osi y zależne od trudności gry (nachylenia toru)

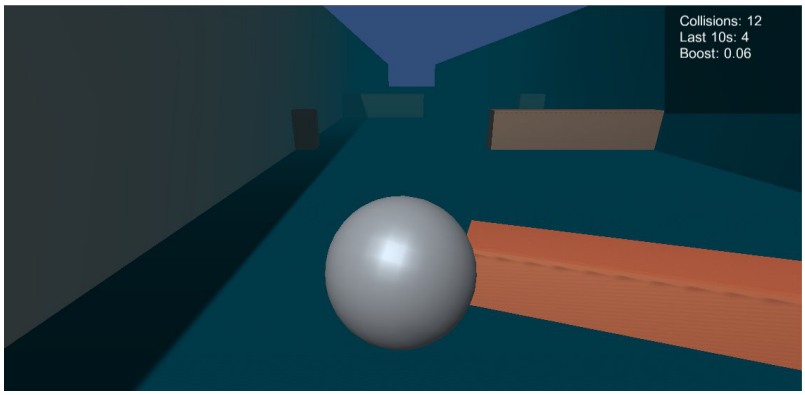
Adam Borowski, nr albumu 137252
Radek Pieczątkiewicz, nr albumu 137373

Dokumentacja wersji BETA

Aktualny stan zaawansowania projektu

Zaimplementowano podstawową wersję gry w trybie jednego poziomu trudności bez rozpoznawania emocji.





Trudności w realizjacji

Trudność w realizacji stanowił algorytm generowania trasy. Przeszkody (tutaj raczej szczeliny) nie powinny być generowane zupełnie losowo. Trasa powinna umożliwiać piłce przechodzenie przez szczeliny mimo jej prędkości i bezwładności. Zaprojektowany algorytm zakłada, że granice szczeliny są losowane ale tylko w pewnym zakresie podobnym do zakresów poprzedniej szczeliny, zależnym od odległości od poprzedniej szczeliny. Dzięki temu trasa nie wydaje się być zupełnie chaotyczna.

Zmiany w projekcie

Ponieważ Unity, jak się dowiedzieliśmy, narzuca pewien sposób organizowania struktury obiektów poprzez przypisywanie klas do obiektów sceny (tutaj MonoBehaviour), uprzednio stworzony diagram potraktowaliśmy jako raczej próbę zrozumienia problemu i nie odwzorowaliśmy go sztywno w kodzie mając na uwadze styl jaki proponuje Unity (niezależne skrypty przypięte do obiektów sceny z możliwością wstrzykiwania zależności z poziomu edytora Unity).
Dodatkowo, wprowadzono kolizje w taki sposób, że ściany przewracają się niczym klocki po odpowiednio silnym uderzeniem kuli. Dodano mgłę odległościową tak, że przeszkody wyłaniają się dopiero będąc w pobliżu gracza.

Uzgodnione odstępstwa od założeń

Uproszczono zależności obiektów gry od emocji / poziomu trudności gry. Trasa gry jest generowana niezależnie od stanu gry. W trybie z emocjami emocje będą rozpoznawane na podstawie przyśpieszenia piłki oraz liczbie kolizji w ostatnich dziesięciu sekundach. Emocje będą wpływały na sterowność i rozmiar piłki. W trybie bez rozpoznawania emocji parametry piłki będą ustalane na podstawie wybranego poziomu trudności w ekranie powitalnym.

Msiball

Adam Borowski, nr albumu 137252
Radek Pieczątkiewicz, nr albumu 137373

Raport końcowy

Stopień realizacji

Zrealizowano najważniejsze funkcje gry, takie jak rozpoznawanie emocji, ustalanie poziomu trudności, tryby gry, naliczanie puntków, kolizje. Dodano dźwięki do gry dla menu, głównej rozgrywki, kolizji i momentu zmiany rozmiaru kuli. Gra jest w pełni funkcjonalna i grywalna. Zaimplementowano również odbijanie piłki na skutek uderzenia w przeszkodę.

Testy wydajnościowe

Gra nie nakładała żadnych wymagań wydajnościowych ze względu na ograniczony rozmiar generowanej sceny. Obiekty, które przestają być widoczne na ekranie zostają utylizowane lub podlegają recyklingowi.

Walidacja w środowisku docelowym

Gra działa poprawnie w docelowym środowisku w systemie Windows 8.1 (64bit). Nie zauważono spadku wydajności w trybie renderowania grafiki najlepszej jakości, w trybie pełnoekranowym.

Odstępstwa od założeń

Ze względu na problemy wynikłe z nieujawnionych zależności kompilatora modułu ER i nowej wersji Unity 5, które powodowało problemy z integracją ERF i Unity, zważając na ograniczone zasoby czasowe ograniczono algorytmy po stronie modułu eksperta ERF. Model emocji wykorzystywał jedno zdarzenie TIME_ELAPSED z dwoma argumentami (liczba kolizji w ostatnich 10s oraz aktualna prędkość piłki) zamiast dwóch zdarzeń z jednym argumentem.

Obserwacje i wnioski

1. Środowisko Unity3D pozwala w szybki i przyjemny sposób tworzyć proste gry 3D.
2. Środowisko Unity3D 5 nie wspiera wystarczająco obsługi błędów podczas ładowania bibliotek DLL. Zaobserwowano zjawisko krytycznego błędu Unity powodującego wyłączenie aplikacji i brak wpisów w logach programu Unity.
3. Ważne jest aby odwzorowania parametrów emocji na zakres kategoryjny (np. poziom łatwy, średni, trudny) był przygotowany w sposób adaptacyjny. Nie można na przykład założyć, że użytkownik nie będzie bardziej przestraszony niż podana explicite wartość. Jeśli na przykład podczas gry emocja strachu przyjmie większą wartość niż założyliśmy, gra musi wziąć pod uwagę ten fakt i zwiększyć zakres który służy do odwzorowania poziomu trudności.
4. Rozpoznawanie emocji gracza jest dużym wyzwaniem, natomiast stanowi dobry kierunek rozwoju gier komputerowych.