Politechnika Świętokrzyska	
Studia Stacjonarne(Semestr Zimowy)2021/2022	
Projekt Programowanie w języku C2	Grupa: 2ID14A
Temat: Gra Rayman/Mario Bros	Zespół: Piotr Równicki Wiktor Sikora

1. Opis projektu:
Gra platformowa wykonana w stylu gier Rayman i Mario Bros przy użyciu języka C/C++ oraz biblioteki Direct2D.

Rayman - seria platformowych gier komputerowych, stworzona przez Michela Ancela i wydana przez francuskie studio Ubisoft. Głównym bohaterem serii jest tytułowy stworek Rayman, a wszystkie jego przygody dzieją się w świecie zwanym Rozdroże Marzeń. Pierwsza gra z serii miała swoją premierę w 1995 roku – była to gra platformowa w grafice 2D.

Mario Bros - komputerowa gra platformowa stworzona przez Nintendo i wydana w 1983 jako gra arcade. W grze mogą uczestniczyć 1 lub 2 osoby (w niektórych wersjach do 4). Gracz pierwszy steruje Mario, natomiast drugi Luigim.

Direct2D – wysokopoziomowe API do programowania aplikacji używających grafiki dwuwymiarowej i wektorowej. Jest dostępny jako część DirectX od wersji 11. Zastąpił zdeprecjonowany interfejs DirectDraw. Direct2D pozwala na sprzętową akcelerację grafiki 2D poprzez kartę graficzną, oferując wysoką jakość i szybkość.

2. Funkcjonalności projektu:
Nasz projekt pozwala umilić wolny czas.
Posiada on:
- sterowanie postacią(lewo, prawo i skok)
- ruchomych wrogów
- możliwość zbierania monet
- piękną grafikę
- dużą mapę

3. Sposób uruchomienia oraz obsługi projektu:
Projekt można uruchomić za pomocą Cmake lub Visual Studio 2019.

4. Stworzone klasy, metody:

Klasy:

Character	Klasa odpowiedzialna za bohatera
CollisionDistances	Struktura odpowiedzialna za dystansu
Enemy	Klasa odpowiadająca za elementy związane z wrogiem
Engine	Klasa odpowiadająca za silnik
EngineBase	Klasa odpowiedzialna za silnik obiektów
GameObjectBase	Klasa odpowiedzialna za baze obiektów gry
HUD	Klasa odpowiedzialna za **HUD** w grze
Level	Klasa odpowiedzialna za level
MainApp	Klasa odpowiedzialna za funkcjonowanie gry
Point2D	Struktura odpowiadająca za punkt 2D

Metody:

Character:

Character()
Konstruktor klasy **Character**.

void **Logic**(double elapsedTime) override
Metoda odpowiedzialna za logike bohatera.

void **Draw**(ID2D1HwndRenderTarget *m_pRenderTarget) override
Metoda odpowiedzialna za narysowanie bohatera.

void **StopFalling**(double collisionSize)
Metoda odpowiedzialna za zatrzymanie spadania.

void **Jump**(bool fullJump)
Metoda odpowiedzialna za skok.

void **BounceTop**()
Metoda odpowiedzialna za odbijanie się

void **StopMovingLeft**(double collisionSize)
Metoda odpowiedzialna za zatrzymanie chodzenia w lewo.

void **StopMovingRight**(double collisionSize)
Metoda odpowiedzialna za zatrzymanie chodzenia w prawo.

void **Die**()
Metoda odpowiedzialna za śmierć

bool **IsDead**()
Metoda odpowiedzialna za sprawdzenie czy nastąpiła śmierć

void **Reset**()
Metoda odpowiedzialna za reset.

CollisionDistances:

void **keepSmallest**()
Funkcja wykrywająca mała odległość

void **keepLargest**()
Funkcja wykrywająca dużą odległość

Enemy:

Enemy(double initialX, double initialY, double maxX, int type)
Konstruktor odpowiadający za stworzenie wroga.

void **Logic**(double elapsedTime) override
Metoda odpowiadająca za funkcjie wroga.

void **Draw**(ID2D1HwndRenderTarget *m_pRenderTarget) override
Metoda odpowiadająca za narysowanie wroga. More...

CollisionDistances **CharacterCollides**(**Character***character)
Metoda odpowiadająca za wykrycie kolizji wroga z graczem.

Engine:

Engine()
Konstruktor klasy opowiadającej za silnik.

~Engine()
Destruktor klasy opowiadającej za silnik.

void **KeyUp**(WPARAM wParam)
Metoda opowiadająca za odciśnięcie przycisku.

void **KeyDown**(WPARAM wParam)
Metoda opowiadająca za wciśnięcie przycisku.

void **Logic**(double elapsedTime) override
Metoda opowiadająca za logikę związaną z silnikiem.

EngineBase:

EngineBase ()
Konstruktor klasy **EngineBase**.

~EngineBase()
Destruktor klasy **EngineBase**.

HRESULT **InitializeD2D**(HWND m_hwnd)
Metoda odpowiedzialna za inicjalizacjie.

void **MousePosition**(int x, int y)
Metoda odpowiedzialna za pobieranie pozycji myszy.

virtual void **KeyUp**(WPARAM wParam)
Metoda odpowiedzialna za odciśniecie klawisza.

virtual void **KeyDown**(WPARAM wParam)
Metoda odpowiedzialna za wciśnięcie klawisza.

virtual void **MouseButtonUp**(bool left, bool right)
Metoda odpowiedzialna za odciśniecie klawisza.

virtual void **MouseButtonDown**(bool left, bool right)
Metoda odpowiedzialna za wciśnięcie klawisza.

void **AddGameObject**(**GameObjectBase***gameObj)
Metoda odpowiedzialna za dodanie objektu.

void **RemoveGameObject**(**GameObjectBase***gameObj)
Metoda odpowiedzialna za usuniecie objektu.

virtual void **Logic**(double elapsedTime)
Metoda odpowiedzialna za logike**EngineBase**.

HRESULT **Draw**()
Metoda odpowiedzialna za rysowanie.

ID2D1Bitmap * **LoadImage**(LPCWSTR imageFile)
Metoda odpowiedzialna za załadowanie obrazka.

GameObjectBase:

virtual void **Logic**(double elapsedTime)
Metoda odpowiedzialna za za logike obiektów.

virtual void **Draw**(ID2D1HwndRenderTarget *m_pRenderTarget)
Metoda odpowiedzialna za za narysowanie objektów.

Point2D GetPosition()
Metoda odpowiedzialna za pobranie pozycji.

HUD:

HUD()
Konstruktor klasy**HUD**.

void **Draw**(ID2D1HwndRenderTarget *m_pRenderTarget) override
Metoda odpowiedzialna za rysowanie elementów**HUD**.

void **AddCoins**(int addCoins)
Metoda odpowiedzialna za dodawanie monet.

void **RemoveLife**()
Metoda odpowiedzialna za usuwanie życia.

bool **HasLives**()
Metoda odpowiedzialna za ilość życia.

void **FinishedLevel**()
Metoda odpowiedzialna za skończenie levelu.

Level:

Level()	
	Konstruktor klasy**Level**.
void	**Draw**(ID2D1HwndRenderTarget *m_pRenderTarget) override
	Metoda odpowiedzialna za narysowanie levelu.
CollisionDistances	**CharacterCollides**(**Character***character)
	Metoda odpowiedzialna za kolizje bohatera z levelem.
int	**PickUpCollectibles**(**Character***character)
	Metoda odpowiedzialna za kolekcjonowanie przedmiotów.
bool	**LevelExit**(**Character** *character)
	Metoda odpowiedzialna zawyjście z levelu.

MainApp:

	MainApp()
	Konstruktor klasy **MainApp**.
	~MainApp()
	Destruktor klasy **MainApp**.
HRESULT	**Initialize**()
	Metoda inicjująca gre.
void	**RunMessageLoop**()
	Metoda pętli głównej.

5. Ilość pracy włożona przez poszczególnych członków zespołu:

Piotr Równicki – 60%
Wiktor Sikora – 40%