[Date]

JAMES MCADAM

Abertay University

James McAdam Course Work Submission

CMP 202 – Data and Algorithms

Table of Contents

[1 Games Design 2](#_Toc508288680)

[1.1 Introduction 2](#_Toc508288681)

[1.2 Target Platform 2](#_Toc508288682)

[1.3 Core Gameplay 2](#_Toc508288683)

[1.4 Other Features 3](#_Toc508288684)

[1.5 Assets 3](#_Toc508288685)

[2 Technical Design (Application Design) 6](#_Toc508288686)

[2.1 Introduction 6](#_Toc508288687)

[2.2 Application Design 6](#_Toc508288688)

[2.2.1 Generic GameObject 6](#_Toc508288689)

[2.2.2 GameLoop 7](#_Toc508288690)

[2.2.3 Vehicle Physics 8](#_Toc508288691)

[2.2.4 Collision Detection 8](#_Toc508288692)

[2.2.5 Rendering 8](#_Toc508288693)

[2.2.6 Audio 8](#_Toc508288694)

[2.2.7 UI / 2D Assets 8](#_Toc508288695)

[2.2.8 AI 8](#_Toc508288696)

[2.2.9 Input 8](#_Toc508288697)

[3 References 8](#_Toc508288698)

# Games Design

## Introduction

This section details the intended game design irrespective of hardware and software consideration in line with design specification (Appendix A). The core elements of the game are broken down and from this an asset list produced. A review of this design was carried out to determine project feasibility. The final scope agreed and an implementation plan produced.

## Target Platform

As per the brief the target platform is the Windows system.

## Core Gameplay

## Other Features

The game will feature the following ancillary features:

## Assets

Based on the initial game design the asset requirements are as follows:

|  |  |  |
| --- | --- | --- |
| **3D MODEL ASSETS (WITH TEXTURES)** | | |
| **Name** | **Description** | **FileType** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **2D SPRITE ASSETS** | | |
| **Name** | **Description** | **FileType** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| **AUDIO ASSETS** | | |
| **Name** | **Description** | **FileType** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# Technical Design (Application Design)

## Introduction

This section will detail how the application will operate with the GEF framework integrating Box2D for the PSVita. Each section will breakdown the problem specification and detail the solution and its predicted impact on the application performance.

## Application Design

The base engine to allow the game development to be constructed on required the following features:

* Generic GameObject
* Game Loop
* Vehicle Physics
* Collision Detection
* Rendering
* Audio
* UI / 2D Assets
* AI
* Input

### Generic GameObject

In order to build the game objects a component based system will be developed based on the standard pattern as presented in Game Programming Patterns (Nystrom, 2014).

In order to achieve the game specification to apply physics, input, collision and rendering the following components were created:

* RigidBodyComponent
* MeshComponent
* SpriteComponent
* AudioSourceComponent
* ControlComponent
* ScriptComponent
* AIComponent

The purpose of each component is to provide the game object with a set of flexible functionality that can be applied to the game object depending on its design purpose. An overview of the function of each component is provided below:

**RigidBodyComponent**

Provides a link to the body within the physics simulation that corresponds to this gameobject. It will also provide a callback for when a collision occurs with this object that can allow custom code to be written for the inherited object update function if required. If this component is attached it will drive the Mesh/Sprite components to update the position based on bodies position within the physics simulation.

**MeshComponent**

Stores the 3D model information, texture information and position/orientation (if no RigidBodyCompent is attached) which will be rendered as part of the Mesh Render call. The z component position of this component will be utilised to determine the draw order.

**SpriteComponent**

Stores a 2D Texture that will be rendered to the screen based on the objects z Position. Again the positon/orientation will be determined by this component (unless a RigidBodyComponent is attached). This will for effects spawned/text/ground to be created in the game. This will contain a UI flag that will enable features such as touch screen button press and ensure this is drawn last after the game (the object z position will still determine the draw order).

**AudioSourceComponent**

If this object generates any sound this component will liaise with the engine to determine the 3D audio element to determine the position of the audio for the user.

**ControlComponent**

The control component will interface commands initiated via the ScriptComponent with a specific gameObject. Its purpose it to design common interfaces for objects that can be driven by a Script or AI component.

**ScriptComponent**

The scriptcomponent’s purpose to create and apply specific logic to the gameobject in question such as physics response, time expire. It can process inputs from the player or triggers from collisions. If the object is a generic controllable type it can call commands from an attached ControlComponent

**AIComponent**

This component can simulate the inputs from a player in order to drive a GameObject. Like the ScriptComponent it will be a friend class too allow it the same access that the ScriptComponet.

The Game Object will have a generic interface that the main gameloop will react with to process and render each of the game objects. The main gameloop will not have any direction interaction with these objects and a number of static functions will exist that will allow gameobjects to Spawn/Destory other objects as required. **Care has been taken to ensure the physics simulation is updated correctly by …**

In order to render and process the objects correctly a list of references will exist based on the dominant component types i.e. Mesh vs Sprite. A separate thread will be responsible for ordering these components if required between frames to allow the main game loop to continue processing the gameObjects. This is a trade-off of performance verses generic design.

### GameLoop

The gameloop will control the state of the game. To achieve this there will be a number of scenes to represent the different game states:

* Splash Screen/Main Menu
* Main Game
* Credits

Each scene will handle the specifics of user input, rendering, physics and gameplay. The game state can be modified by the scenes and the gameloop will handle the transition by overlaying a loading screen prior to any processing taking place.

### Vehicle Physics

In order to create the racing game from the top down perspective the box 2D physics was configured with no gravitational effects. This resulted in the x and y coordinate being equivalent east and north. Initial design of this mode would be based on a tutorial from Engineering.Net (Engineering.Net, 2010).

The aim is to simulate the effects of the tires on the 2d body and apply the necessary forces to achieve the desired driving effect. This was initially achieved using the bicycle model where a front rotatable tyre and a rear fixed tyre are modelled (Engineering.Net, 2010). **(Temp)**

A simplified car model would be used based on tutorials from iforce2d (iforce2d, 2014) and Radu Angelescu (Angelescu, 2015).

The key considerations for the physics are:

* Friction
* Driving forces
* Turning torque

**Friction**

The friction will oppose any lateral acceleration of the car. This will have an upper limit that once overcome will result in the vehicle beginning to skid. There will also be a wheel rolling friction that will result in the vehicle decelerating when no driving force is applied. There is also a braking force applied to the wheels. Again this will have a threshold that once exceeded will result in the vehicle skidding with an inability to turn. (Kinetic friction is lower than static) i.e. a wheel skid slows slower.

**Driving forces**

The vehicles will be able to apply a forward acceleration and a reverse function.

**Turning Torque**

Turning the vehicle will apply a torque to the RigidBody Component to rotate it.

### Collision Detection

Implement a call back system from the rigidbody which will call a function called collision which passes a pointer to the object being hit for query.

### Rendering

A separate thread will order a list of pointers to either mesh or sprite objects to ensure they are correctly ordered. In the event of a new object being rendered the thread will be tasked with inserting it into the correct order.

### Audio

This will operator on event system which passes the sound and the location. The world space is calculated and the sound produced based on its 3d location.

### UI / 2D Assets

These assets will be drawn last to create a HUD system. They will have a specific order and if enabled will have interaction with the input control system to detect if a screen touch has initiated them.

### AI

The AI will need to be individual and respond realistically to the track situation in order to be believable. It will include obstacle avoidance and optimum track behaviour. For the sake of this demo the AI will be given at random, al be it rarely, power ups to use and will use them instantly. The focus will be on making opponents respond correctly.

### Input

An input singleton will handle all input and make it available to all gameObjects as required.

# References

Angelescu, R., 2015. *Top down racing car physics model using box2D.* [Online]   
Available at: http://www.raduangelescu.com/topdownracingcarphysicsmodelbox2d.html  
[Accessed 8 March 2018].

Engineering.Net, 2010. *Simple 2D car steering physics in games.* [Online]   
Available at: http://engineeringdotnet.blogspot.co.uk/2010/04/simple-2d-car-physics-in-games.html  
[Accessed 8 March 2018].

iforce2d, 2014. *Box2D C++ tutorials - Top-down car physics.* [Online]   
Available at: http://www.iforce2d.net/b2dtut/top-down-car  
[Accessed 8 March 2018].

Ign, 2009. *IGN.* [Online]   
Available at: http://uk.ign.com/games/grand-theft-auto-chinatown-wars  
[Accessed 5th March 2018].

Nystrom, R., 2014. *Game Programming Patterns.* 1st ed. Marston Gate: Amazon.co.uk Ltd.

Wikipedia, N/A. *Grand Theft Auto (video game).* [Online]   
Available at: https://en.wikipedia.org/wiki/Grand\_Theft\_Auto\_(video\_game)  
[Accessed 5th March 2018].

Appendix A – Design Specification