International Journal of Research in Engineering, Science and Management
Volume-2, Issue-3, March-2019
www.ijresm.com | ISSN (Online): 2581-5792

PC based Game Design

Samayranjan Pradhan[1], Shubham Sawant[2], Suraj Nishad[3], Sanket Chaudhary[4], Nilesh Kulal[5]
[1,2,3,4]*B.E. Student, Department of Computer Engineering, Terna Engineering College, Mumbai, India*
[5]*Assistant Professor, Department of Computer Engineering, Terna Engineering College, Mumbai, India*

***Abstract*: Computer gaming is a key component of the rapidly growing entertainment industry. While building computer games has typically been a commercial endeavor, we believe that designing and constructing a computer game is also a useful activity for educating one about geometric modeling and computer graphics. In particular, we will be exposed to the practical issues surrounding topics such as geometric modeling, rendering, collision detection, character animation and graphical design. There are plenty of frameworks available on the internet for game development, some are free (Cocos2d-x, Godot) while others have proprietary-versions (Unity, CryEngine, Unreal), but the real issue is big guns of gaming industry developing their own game engines for eg. Rockstar and Ubisoft using RAGE and AnvilNext respectively, this makes it difficult for a developer to go out of its comfort zone and learn an entirely new framework. Hence, the goal of the project is to create a 2D game framework using box2D and UI for basic game design. Although, the end-result is a generic framework for desktop games a use case of racing game will be taken.**

***Keywords*: Game; Game Framework; 2D Game**

1. Introduction

This research paper will give you the gather and analyze and give in-depth insights of the PC based Game Design framework system. The Hardware/Software requirements along with the Functional/Non-Functional requirements are properly specified in this document. The purpose is to develop a 2D game framework, which will contain all the tools for game development. This paper focuses on the core framework and its application in a real world use case. Among its current features, there are:

- Physics support through the Box2D library.
- Static and animated layers, including parallax scrolling.
- Support to state-based sprite animations.

The project roadmap also includes support for background music, audio effects, networking and cut scenes.

2. History

Before game engines, games were typically written as singular entities: a game for the Atari 2600, for example, had to be designed from the bottom up to make optimal use of the display hardware—this core display routine is today called the kernel by retro developers. Other platforms had more leeway, but even when the display was not a concern, memory constraints usually sabotaged attempts to create the data-heavy design that an engine need. Even on more accommodating platforms, very little could be reused between games. The rapid advance of arcade hardware which was the leading edge of the market at the time meant that most of the code would have to be thrown out afterwards anyway, as later generations of games would use completely different game designs that took advantage of extra resources. Thus, most game designs through the 1980s were designed through a hard-coded rule set with a small number of levels and graphics data. Since the golden age of arcade video games, it became common for video game companies to develop in-house game engines for use with first party software.

In the 1990s, there was several 2D game creation systems produced in the 1980s for independent video game development. These include Pinball Construction Set (1983), Arcade Game Construction Kit (1988).

3. Architecture

The purpose of the document is to gather and analyze and give in-depth insights of the PC based Game Design framework system. The Hardware/Software requirements along with the Functional/Non-Functional requirements are properly specified in this document. The purpose of this project is to develop a 2D game framework, which will contain all the tools for game development.

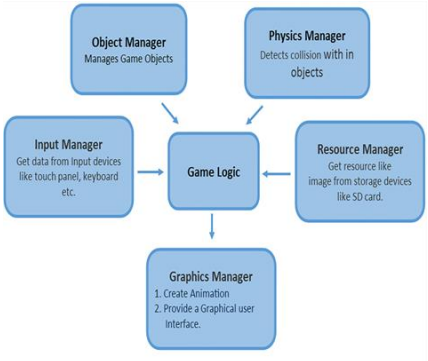

Fig. 1. Architecture

We have divided the Game Engine into 5 components. We will address these components as managers, responsible for performing some specific tasks.

4. Components

To understand the working of the architecture the following components required

A. *Input Manager*

The term "input" or "user input" refers to the data that a player provides at run time to a game using an input device. The structure and type of the data varies from device to device. For example:

- A touch panel provides information of the coordinate (x, y) being touched.
- A PS2 mouse provides data in the form of packets which contains the button state and mouse movement (x, y) information.

Input manager provides an abstraction layer to access input devices in a system. It calls functions of a hardware dependent input library for the specific device to get input data. So, if we write our game code using the Input manager with keypad as an input, and later we want to use a joystick as an input device instead of keypad, then we will be able to do it without major changes in the code. Also, the game will be portable across different platforms.

B. *Physics Manager*

Physics Manager performs two important tasks:

- Simulate movement: It simulates the movement of objects (or characters) like walking, running or jumping. Physics Manager simulates forces like gravity, friction to generate such effects. For example, in the Mario game, when the Mario is moving with a velocity (Vx) and the player applies an upward force on the Mario to make him jump using the keyboard, the Mario rises in the air and over time, the force of gravity will act against that initial jumping force, which will give that nice classic parabolic jump pattern.

C. *Collision detection:*

It detects intersection of two or more objects in a game. For example, in the Mario game, the Mario can collide with bricks, bullets and enemy objects. The scope of this topic is very vast. However, we will restrict our discussion to basic collision detection only. The objects are restricted to 2D space and so, we will be focusing on three basic types of collisions:

- Circle-Circle Collision
- Rectangle-Rectangle Collision
- Rectangle-Circle Collision

D. *Object Manager:*

An object is the smallest entity in a game that a player can see and/or interact with. The character, enemies, weapons, bullets and background elements such as trees and clouds are all game objects. The game logic is written around these objects and the managers perform various tasks on these objects. The state of an object at any given time is defined by its attributes. For example, state of the ball in the Arkanoid game is described by its position (x, y) on the screen and its velocity (speed and direction of travel).

- *Static:* An object which is stationary throughout the game.

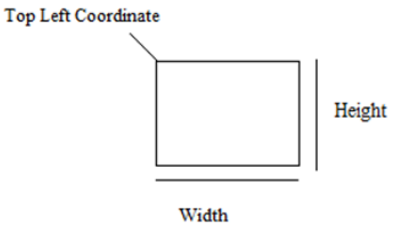

Fig. 2. Static

- *Dynamic:* An object which can move in a game. For example, cars in a racing game, ball in the Arkanoid game etc.

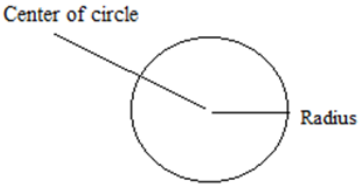

Fig. 2. Dynamic

E. *Resource manager*

There can be various types of resources in different formats as listed in Table.

Table 1
Resource

Resource	File Format	Use
Image	BMP, JPG, JPEG,GIF etc.	Background music, play tunes on some action, character voice
Sound	Buzzer sound, MP3, AAC, WAV etc.	Game information, game instructions, developer information. All such information can be stored in a text file and displayed in a game.
Text	TXT Files etc.	Game information, game instructions, developer information. All such information can be stored in a text file and displayed in a game.

F. *Graphics manager*

Graphics is a vital part in any game. Graphics display techniques have evolved over time from both hardware and software perspectives. Graphics works by appealing to our senses, by drawing our attention and holding on to it and by immersing us in their world. Graphics manager in our Game Engine is responsible for handling graphics. It also provides a graphical user interface (GUI) module which can be used to create a menu window, help window, game controls window etc.

Graphics manager can be divided into two modules:

Game Graphics: It is responsible for handling graphics in a

game. It creates graphics according to different shapes of an object such as circle or rectangle. For example, to create a motion effect for an object and move the ball along the X axis, Graphics manager will perform the following steps:

- Draw a circle at position (X, Y) on the LCD screen.
- Draw a circle of same radius with background color at (X, Y).
- Draw the circle at new position (X+1, Y).
- Graphical User Interface: Graphical User Interface or GUI is a type of interface that allows users to interact with an application program through graphical elements such as menus, widgets and dialog boxes. GUI elements are usually accessed through a pointing device such as a mouse or a stylus.

5. Future scope

The prospects are huge, now-a-days gaming is coming like anything. There are lot of scope over here like Gaming is now used at Bank's, for marketing, for education, to improve the IQ etc. Mobile gaming, i-Phone Gaming and Social Gaming is the next gen future. One can start his or her career as a gaming artist, game developer, game taster, game de-coder, porting etc. Another type of gaming is coming with lot of scope, X-Box, Nintendo and Simulation Games.

6. Conclusion

Hence, the project report on the proposed system has been successfully drafted. The proposed system offers a 2D lightweight game framework, which could be used by game developers or by game enthusiasts for developing simple 2D games. The proposed system strives to be platform independent and easy to use/understand. A simple use-case of car racing pc game will be taken.

References

[1] S´ergio Correia, Rodrigo Gonc¸alves de Oliveira, Roger Zanonifsergio. correia, rodrigo. goncalves, "Developing2D games in a declarative way, Proceedings of SB Games, pp.33-36, 2012.
[2] Florian Knoll, Box2D C++-Top-down car physic, iforce2d_Top down Car Race Track.h, March 2014.
[3] David Brackeen, Bret Barker - Developing Games in Java 1st edition.
[4] "D Box2d working, http://www.iforce2d.net/b2dtut/