INTERNSHIP REPORT

CSC410

Abstract

This report is being presented as part of CSC410 course to describe the details of the internship done as part of the summer internship process of the IIT-Delhi curriculum.

Shantanu Chaudhary

2010CS50295

Internship at Future Captcha PVT. LTD, Gurgaon

About the company:

The company is a start-up founded by Mr Lokesh Jaiswal in November, 2012. The company works in the domain of game development and electronic commerce.

Company Division:

The company has two divisions:

1. Game Development Studio
2. Innovation Studio

The game development studio carries out various projects of developing games for different platforms such as android, IOS, windows.

The innovation studio is responsible for development of an e-commerce website. Their first venture is a food ordering website for cheap and convenient delivery of food to customers.

Tools used:

- SVN: For version control
- Unity3D: SDK for development of games
- Mono Develop: Editor for programming behavioural scripts
- 3DS Max: For modelling characters and game objects
- Photoshop: For making GUI Textures
- Unity Script/JavaScript: For programming behaviour of game objects

Working methodology

The development model used as part of project development was the *Evolutionary Development Model (Software Engineering)*. Under this model, the product (game) is divided into a set of features. Basic game is developed and released in the marketplace and other features are added gradually over time. These features are developed as part of weekly sprints. Two sprints would be held every week (Sprint 1: Mon-Wed & Sprint 2: Thur-Sat). On completion of a sprint the developed features would be checked for bugs. After this check, a platform (android, IOS, windows) build for the project would be presented to the Chief Technical Officer for approval. After approval, the developed feature would be merged with the master build in the project repository, tagged with appropriate version and then it would be deployed in the market place of respective platforms (Google Play store).

About the Projects

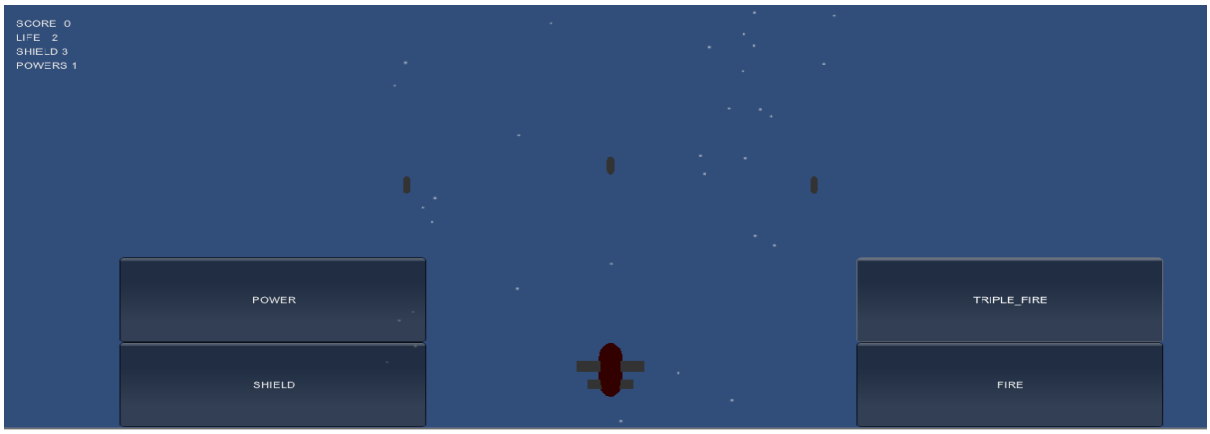
We were required to work on the following games as part of our internship at the game studio.

Training games:

These games were assigned to us as individual exercises so that we are able to learn about the tools we would use in the development of the main projects. These games were built for android platform. These included development of 2D and 3D games with touch interface:

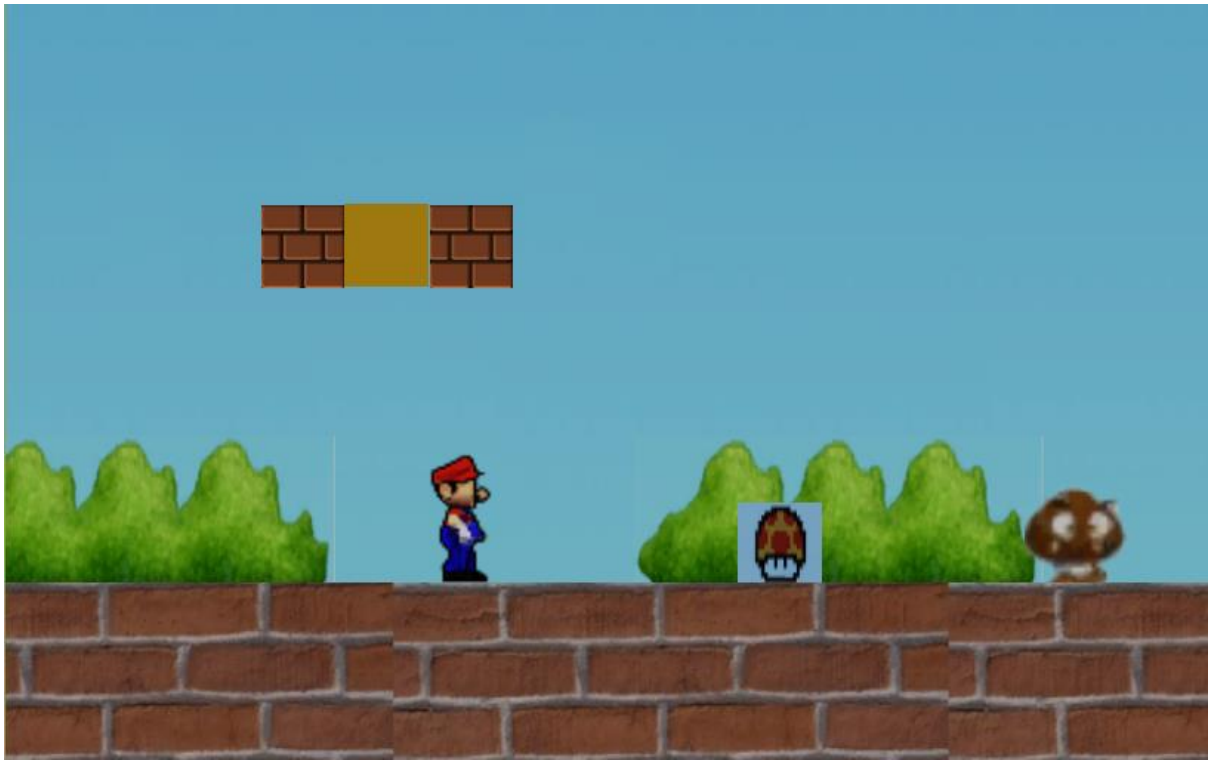
Screen shot of Fruit Ninja spoof

- Fruit Ninja Spoof- This game was just a spoof of the famous android game Fruit Ninja. By developing this game as an exercise we learnt to deal with rigid body and how it is handled by the Unity engine.



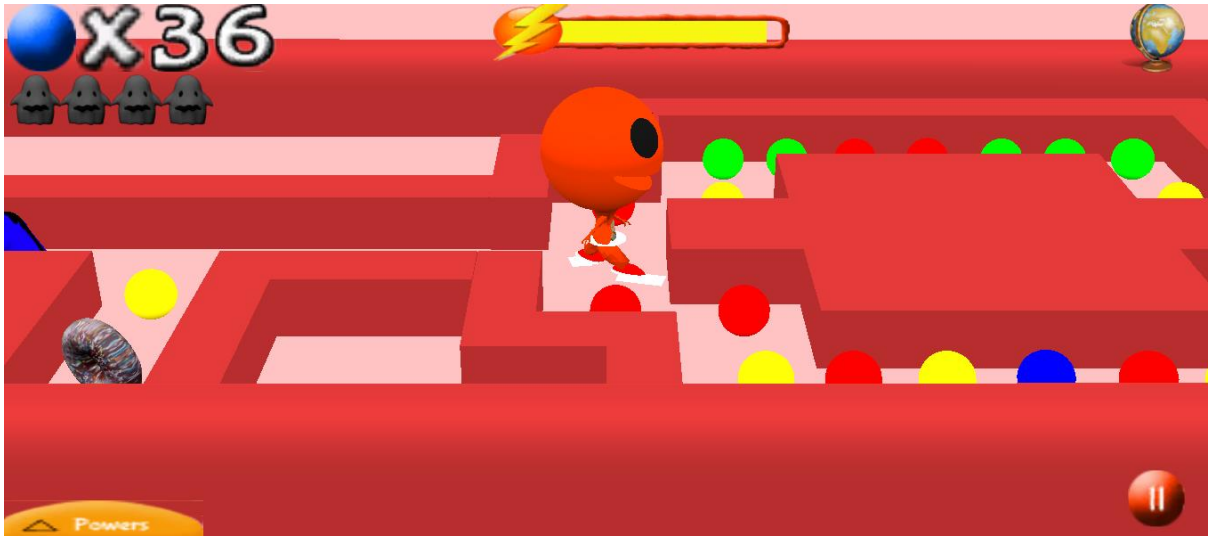
Screen shot of Space Fighter

- Space Fighter- This was actually a 2D aeroplane game flying through a star field. Through this game we learnt about generating particle effect (to simulate the effect of falling stars) and how to simulate fire and blasts and maintain control over powers of the player.



Screen shot of Mario 2D

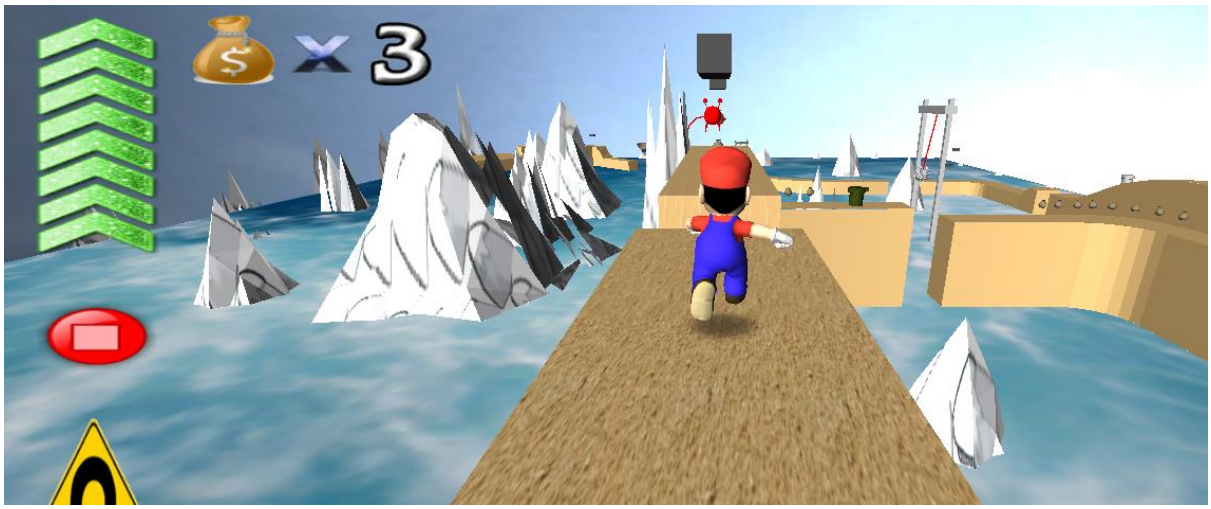
- 2D Mario- This was a spoof of the classic Nintendo Mario game. This game was our first exercise as a team. This game taught us the essentials of 2D animation (sprite sheet animation). We as a team of 3 developers divided the development of this game into 3 parts and developed these parts independently and then integrated these parts to make the final build.



Screen shot of 6Packman

6-Packman:

This game is just a 3D variant of the classic 2D game PACMAN. The gameplay involves the player running through a maze tackling enemies and picking up collectibles so as to attain a certain high score and clear the level. The game had time attack and arcade modes so as to offer a variety to the user.



Screen shot of Mareon

Mareon:

This game is the 3D variant of Nintendo's Super Mario Brothers. This project threw all sorts of challenges at the developers such as 3D animation, optimised rendering of world elements, efficient and easy gameplay controls for the player based on touch input and many more. The special feature which this game offered was that the world elements in the game had been derived from real

world elements (such as continents, mountains, monuments and historic places).

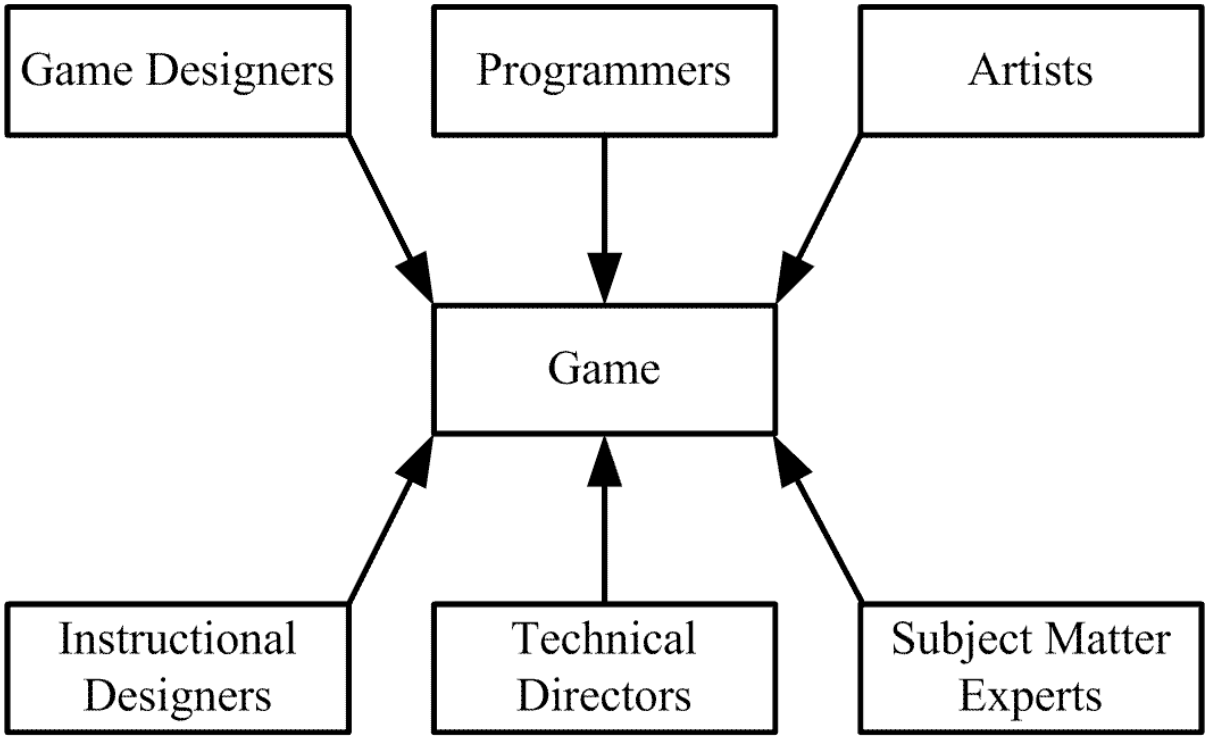
Screen shot of Down the Hatch

Down the Hatch:

This game is a story of a micro-organism which is flushed down a drain and comes across organisms which are larger than it and pose a danger to its existence. The player must tackle the enemies and consume certain collectibles which would enable the player to kill enemy organisms and attain certain score.

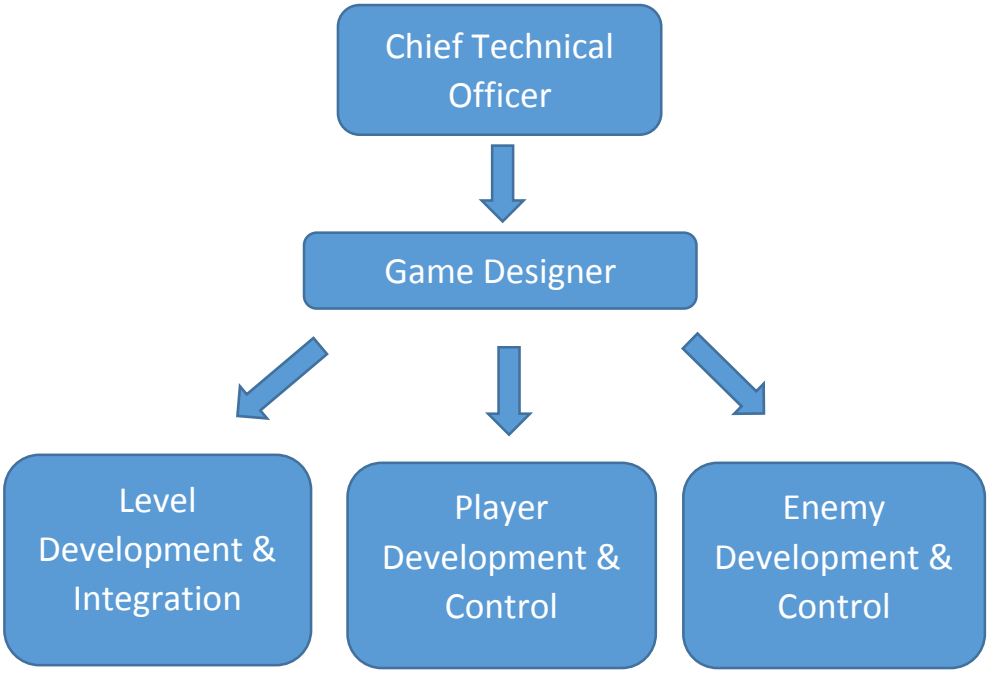
About my role

The team of the game development was part of a hierarchy described by the following diagram:



Since this company was in its start-up phase, we as interns had to assume multiple roles of designing gameplay, debugging as well as GUI development.

Broadly our team consisted of the following people:



About my work

I worked as a developer responsible for level development and integration in the team. Under this designation, I was required to work on the following features:

- Developing world elements such as terrain, platforms, collectibles, animations in the world, and stimuli for player interaction.
- The development had to be as modular as possible so as to support easy integration with the work of other developers.
- Moreover the elements should be scalable so that they can be used in other levels. Hence they should offer scalability when the project scales.
- Using finite state machine models to maintain pause/resume states of the game play scene.
- Maintain parameters such as health, score, and powers of the player by means of global scene scripts.
- Manipulate camera view as required by game designer. (First person view or Third person view)
- Optimising performance of game for mobile devices such as checking tearing of textures, calibrating anisotropy, culling game objects in order to reduce rendering load etc.

6Packman:

The main challenge faced during the development of this game was to develop game objects in such a way so that they can be used in a modular way. This would be helpful when we would add more stages to the game. Hence the player control scripts, collectibles objects, and behavioural scripts were generic. The important thing we learnt in this developmental exercise is that we came to know about the notion of "*prefabs*". For example, if we had a collectible power up, we could frame it as a prefab and then simply place it in the game scene or any other scenes we wanted it to be in. Unity SDK provides sophisticated tools which allows the developers to handle the instantiation as well as destruction of these prefab game objects. So to sum up, all the labour is applied in designing the role of different game objects, generating their prefabs and then developing world by placing them in the scene.

Another important challenge was to develop an AI for enemies. As described before, during the gameplay, the player has to make way through the maze and

tackle enemy. At the same time, the enemy should be intelligent enough to determine the position of player and is able to catch the player. Various strategies were used to develop an AI for the enemy ghosts. We worked on the following two options:

1. Greedy algorithm: This algorithm queried the position of the player and determined the path to the player by geometry and knowledge of maze.
2. A* Algorithm: This approach provided us with exploiting every possible best path to the player and choosing the right path from a set of options arranged in the form of a tree.

Another important aspect we worked on is saving data to the device filesystem in order to track the progress of game. Important user information had to be stored onto the device filesystem (with encryption) whenever the game reached crucial points in the game. Unity provides "*PlayerPrefs()*" calls in order to save certain integers and strings to the mobile device file system. On starting the game application, the information is read from the filesystem during loading stage.

Mareon:

In this game, most of the challenges were similar to previous game project but a new challenge in this project was to make a strict path following character. That is the player character must stick to the designated path in the free world.

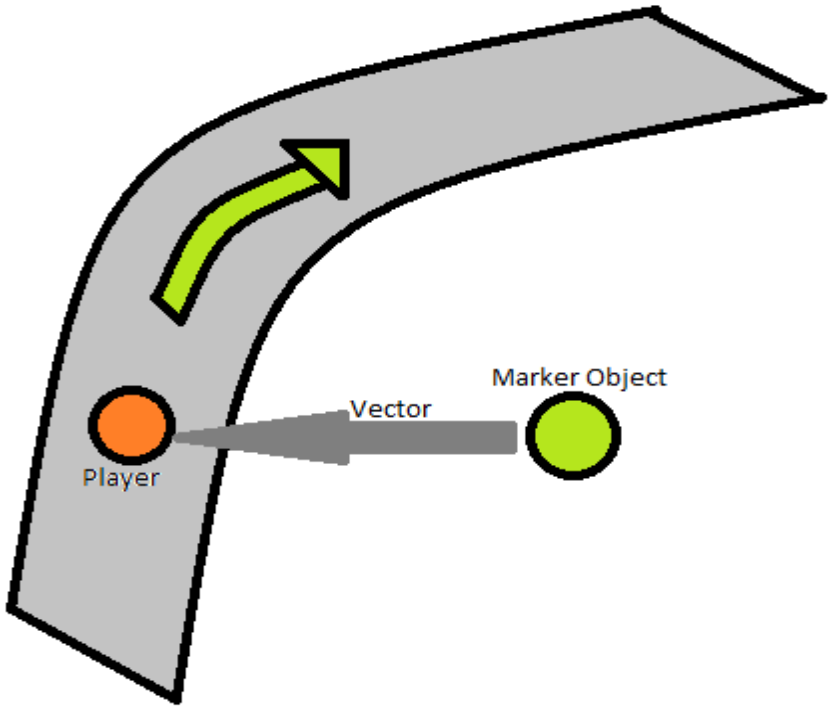
That meant whenever the path turned right or left, the character should also turn right or left respectively. And these turns were not perfect 90 degree turns, there were all sorts of curved paths all over the terrain. To make the character translate in a linear direction was easy but to turn the character such that it sticks to the path was the challenge we faced. So we researched different unity forums to come with a scheme that would enable us to achieve our aim. After carefully researching out the forums we came up with 2 approaches:

1) Waypoint approach

2) By use of vectors

In the waypoint approach, we place planes along the path which act as checkpoints. As the player passes through them, a signal is triggered which keeps the player on the path. This scheme is effective and accurate but the problem with this scheme is that if we place such waypoints along the curves of the whole path then the rendering of the game on mobile devices would be slowed down ruining the gaming experience.

The use of vectors eliminated the use of waypoints. Now we had to take care of only the curved paths and the turns. In this approach, when we want the character to walk/run around a certain curve, we find the radius of curvature of the curved path, place a marker object at the centre identified by the radius of curvature. Now when the character comes across the curved path we extend a vector from the character to the marker object and then rotate that vector around the object in a specific region along the curve. This strategy perfectly executes the curve of any curvature.



Optimisations:

Both the projects required significant optimisations in terms of build size, rendering, texture quality, gaming experience. We tested our projects on devices with different CPU/GPU specs. We tested our builds on Sony Xperia Z cell phone, Sony Xperia Neo cell phone and Samsung Galaxy tab 800. The aim was to test texture and render quality for different devices. Based upon the performance, we applied camera culling so as to minimise the rendering load on the CPU/GPU. We also optimised textures by calibrating their anisotropy and level of detail which in turn reduced the size of build package. Another important lesson we learnt was to optimise the game display for devices with varying display resolutions. Our games were optimised for Full HD displays with Qualcomm Adreno GPUs. Thorough effort was put to minimise variations across devices.