2016

Andrew Davis

“Movies and Shakers”

Design Document

GAME REPORT – Andrew Davis

Contents

[1 Basic Concept 2](#_Toc445480907)

[2 Design Details 3](#_Toc445480908)

[3 Software and Hardware 6](#_Toc445480909)

[4 Graphics and Audio 8](#_Toc445480910)

[5 Code Structure 10](#_Toc445480911)

[6 Game Specific Features 14](#_Toc445480912)

[7 Time Management Plan 16](#_Toc445480913)

[8 Bibliography 18](#_Toc445480914)

**APPENDIX A - GRAPHICS:**

[1 Staff 19](#_Toc445480915)

[2 Customers 19](#_Toc445480916)

[3 Screens 20](#_Toc445480917)

[4 Other Purchasable Items 21](#_Toc445480918)

[5 Buttons 21](#_Toc445480919)

[6 Currencies 21](#_Toc445480920)

# Basic Concept

* My game is going to be called **“Movies and Shakers”**
* The genre of my game is going to be **Management and Simulation**
* The game will be targeted to the **iOS platform – iPhone, iPad, iPod touch**
* The target audience for my game will be players of both genders, mainly aged **10-20 years old**. However, I hope that my game will appeal to other age groups as well. A common feature of mobile games of the genre that I am creating is that they should allow the player to have a ‘quick go’ and then reach a point where they can stop and pick up again at a later point. Hence, I also hope to target **commuters** (on busses/trains) who are looking for a time filler on their journey.
* The game will use two main Game Mechanics: **Worker Placement** in regards to the gameplay where you designate staff to various tasks, as well as **Resource Management** which relates to the earning and spending of the two currencies in my game – coins and popcorn. These ‘resources’ can then be exchanged for other resources – such as new screens and decorations.
* The unique selling points of my game are that it gives an insight into how a cinema operates and also allows you to create a cinema the way you want to – customizing everything from the carpet colour to the layout of the screens and even which staff to hire!
* There are a couple of similar games to mine on the market, such as: *“FarmVille”, “CityVille”* and *“The Simpsons – Tapped Out”.* However, there aren’t (as far as I can find), any games which allow you to build a cinema like my game.
* The basic idea of my game is:
* The player starts off with an almost empty cinema. They then have to earn money and use the money they earn to build new screens and objects and upgrade the objects. The more screens (and the more upgraded they become), the more customers will show up to the cinema for every screening.
* As well as buying, upgrading and repositioning items, the player will have to manage their staff wisely. There will be several different positions where staff can work – including selling tickets, and showing customers to their seats – so the player will have to move staff to areas where they think help is needed most! If they make good choices and they use their staff efficiently, they will serve more customers and hence boost the reputation of their cinema – enticing more customers to future shows. Staff can also be trained to make them deal with queues more quickly, as well as boosting other attributes which effect the reputation of the cinema.
* There will be a lot of customisability in the game as the player will have complete control over where to position the screens, which objects to buy/upgrade and when, even the carpet colour can be changed!
* There is no real end point to the game – the aim is to develop as successful a cinema as possible!

# Design Details

Starting Point

When a player starts a new game, they will begin with an almost empty cinema building – the only things that will be there will be a single screen (at the lowest upgrade level (level 1)), a very basic ticket booth (probably just a table), and two staff members (both of whom will start with all attributes at the lowest point). They will start with a small amount of each of the two currencies in the game – coins and popcorn (the amount awarded of each currency will be decided once I figure out the cost of objects in the game). With all objects in the game, the player has the option to move them around on the screen, using a tile-based system. The only object that cannot be moved is the ticket booth, which I have decided to lock to one location in the middle. Construction will not be allowed in front of the ticket booth (to allow there to always be a clear path from the front door to the ticket booth).

There is also the option to load a previously saved game so the player can just pick up where they left off last time they played the game. This option will be available in the main menu – the first screen the player comes to when they run the game. The game will ‘auto-save’ at the end of each ‘business day’, so if (for whatever reason) the game was to close unexpectedly at any point, then the player would be able to continue from the last day that was completed.

Gameplay

The actual, serious gameplay will start once the player clicks ‘Begin Day’. This will trigger the start of the ‘Business Day’ and customers will start to arrive at the cinema to see their film. Each customer will have different routes through the cinema to suit their needs. For example, some customers will need to go to the ticket booth to purchase a ticket while others have pre-booked so can bypass the ticket booth. Or some customers may want to buy food while others do not. So the customers will go on their own routes, visiting the necessary places.

The coins earnt for the day will be collected at the end of the day. The shop and other upgrade options are not available while a business day is in progress.

Each customer will have a ‘Customer Satisfaction Level’. Every time the customer has to wait in a queue or is unhappy about something, this level will drop. If the level falls below 0, the customer will run out of patience with the cinema and leave, and the money they would have spent is lost.

Currencies

Coins will be the main currency in the game. They will be used for purchasing items from the shop and performing upgrades on various items. The ‘popcorn’ currency will be the secondary currency. It will be a lot harder to collect popcorn than it is for coins and so it will be used for ‘luxury’ items in the shop, and it will also be possible to spend popcorn to finish construction more quickly. Each customer will spend money at the cinema which can then be re-invested in the shop to improve the cinema further and thus entice more customers back.

Shop

The ‘Resource Management’ game mechanic is present in the shop feature. The player will be able to use their acquired coins and popcorn to purchase several items that will improve their cinema. The biggest increase in the number of customers who will attend will come from purchasing a new screen. This will mean that there are more films shown per day and hence more customers will show up, bringing in more money. There will also be more items available from the shop which can help bring in more money – a food court will be purchasable which will provide food to the customers, and in will result in more money for the player.

Not all items in the shop will increase the daily amounts of money generated. There will also be decorations such as statues and plants which the player can purchase to customise their cinema and make it look the way they want. There will also be an option to change the colour of the carpet – allowing for even more customisability.

Advertisement items will be available in the shop as well. These will include things such as posters, signs and leaflet stands which will increase the reputation of the cinema and entice more customers to visit.

There will also be the option to hire more staff which can help to improve the speed of service and hence, also increase the reputation of the cinema. The name and appearance of the staff can be changed if the player desires.

Staff

The staff members are essential to the smooth running of the cinema. This is where the ‘Worker Placement’ game mechanic comes into effect. The player must choose where each staff member is ‘posted to’, i.e. which job they have, in order to try and maximize efficiency. For example, there will be a slot for a staff member at the ticket booth (multiple slots if the ticket booth is upgraded) and the staff member here has to sell tickets to the customers before the customer can get through to their screen. The player must choose where to position each staff member in order to keep wait times to a minimum. The movement of the staff member will be controlled with the touch screen ‘drag’ control of the device. When the player starts dragging a staff member, all the available ‘slots’ into which the staff member can be dropped will appear. When a staff member

Like screens, staff are also upgradable but in a slightly different way. Each staff will have a number of attributes: ‘Ticket Speed’, ‘Food Serving Speed’, ‘Friendliness’ for example. These attributes can be upgraded individually and will affect the jobs which require them. e.g. to use the ticket selling example again, the higher the staff member’s ‘Ticket Speed’ is, the faster customers will be served, and the happier the customers will be.

Staff members can be repositioned during the business day, so if the player notices that a big queue is forming at a certain point, they can move a staff member to help out at that post. This aspect of the game is based on a real cinema/theatre workplace environment where staff often have multiple jobs during the same shift.

Upgrades

A few of the purchasable object types will be upgradable, the screens and the food court amongst others. Upgrading the screen will allow more tickets to be sold for screenings in that screen, while upgrading the food court will allow more variety of food and drinks to be sold – causing more customers to purchase food. A graphical representation of how the screens will be upgraded is available in Appendix A: Graphics.

Reputation

The player’s cinema will have a reputation which will affect the number of customers who attend film showings. This reputation will be composed of several different components – such as ‘speed of service’, ‘cleanliness’ etc. I will think of some algorithm which will create a multiplier based on the reputation, which will then be applied to the number of people who attend.

Posters, leaflet stands etc. will help to increase awareness of the cinema.

Menu

The menu will be along the bottom of the device’s screen and will display all the options for editing staff members, an option to visit the shop and the option to begin a day. It will also display the current day, current time, and how much of each currency the player has. The player will be able to hide the menu if they so wish.

Additional Features

There will be smaller features in my game which contribute to the overall reputation of the cinema, but are out with the standard game play. For example, one such feature could be that a bin is overflowing and the player has to spot this and designate a staff member to empty it. If they take too long, it will negatively affect the ‘cleanliness’ portion of the cinemas reputation. I hope that by adding this feature, and similar small features, it will keep the

Social

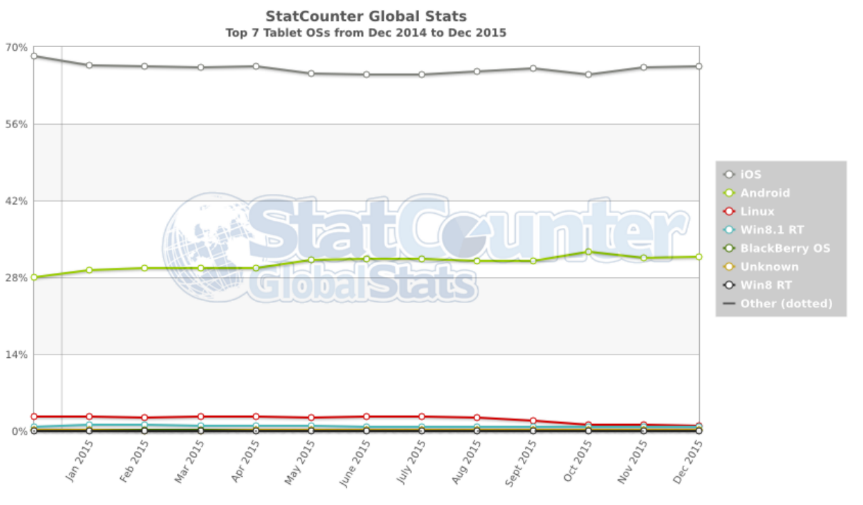
Players will be able to log in and save the state of their cinema with Facebook. As well as this, they will be able to view the cinemas of their friends to have a ‘look around’ – they will not be able to alter their friends cinema in any way – i.e. they will not be able to move the objects or staff, and will not be able to start a business day.

The Facebook player’s basic profile information may be used in the game to tailor the appearance of certain (minor) game elements to them – i.e. a movie poster with their profile picture on it. I hope this feature would be humourous to the player. The player will also be able to send gifts to their Facebook friends.

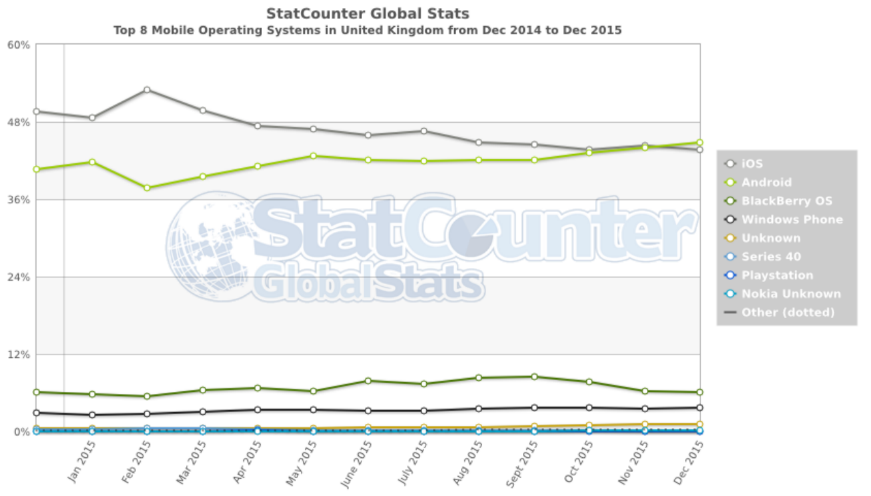
# Software and Hardware

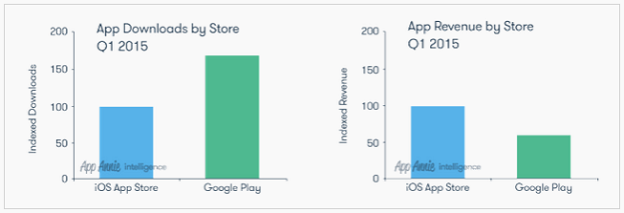
I decided to develop a mobile game because it was an area which interested me. I am not a massive gamer myself but do play console and mobile games occasionally. However, when it came to deciding a platform, developing a console game did not appeal to me so I decided to go for a mobile game. The type of game I had planned to make (Management Simulation) tends to be a ‘casual’ type of gaming, hence is very suited to a mobile device.

I was then faced with a choice which I’m sure many people have faced before: which mobile operating system to develop for? I decided to develop a game for iOS because it is one of the most popular platforms on the market. A study by *StatCounter* showed that iOS was (throughout the year of 2015) by far the most popular Operating System for Tablet devices, with over 65% of Tablet devices using it.



The graph shows that iOS is comfortably the most popular iOS for tablet worldwide and that the nearest rival (Android) is comfortably behind, with only around 30-35% of tablet devices using Android. And there is no sign that iOS will be replaced at the top any time soon.

In terms of mobile phone operating systems in the United Kingdom, there was somewhat of a shift in power in 2015. The same study from *StatCounter* found that at the beginning of the year, iOS was the most popular device. However, by the end of the year, Android had edged ahead in the rankings. However, I decided to target iOS for my game because overall (combining the figures for Tablet and Mobile), iOS is still more widely used. My game will not just be targeted towards the United Kingdom, but will be suitable for use in any country, as going to the cinema is not just a UK past time – almost every country in the world will have cinemas.

As well as the number of users that each platform had, I also did some research into how apps do on the respective markets for Android and iOS. I intend to release my game on the market after this project is complete so I was interested to see which store was more popular for downloading apps. A study from *App Annie Intelligence* found that, although more apps are downloaded from Android’s Google Play store, apps on the iOS App Store tended to have a higher revenue. Since I am interested in putting my App onto the market, this is a figure which helped with my decision of choosing iOS as my platform.

Having chosen iOS, I then had to decide on what software I would use to develop my game. I decided that I would need an engine to make several aspects of the game, such as handling how the customers are created and move, a lot easier for myself, as well as making the implementation of a ‘Tile-Based’ floor easier.

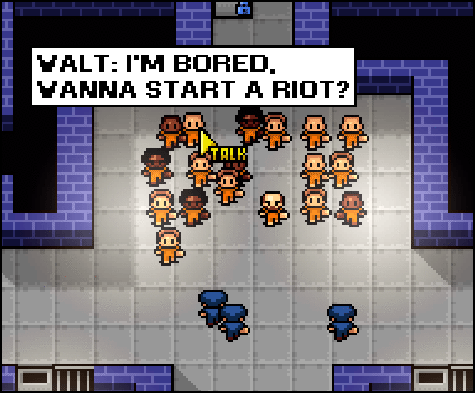
In the end, I decided to use the Unity engine.

One reason for choosing Unity is that the Scripts to control the game can be written in C# which is a language which I am comfortable in and have been successful with before. So I would not need to learn a new language and would be able to just dive straight into the coding when I was ready.

As well as this, I realised that I would have trouble with creating a game for iOS since I would primarily be developing it on my laptop which uses Windows – and Apple only allow apps for their devices to be developed on other Apple products, and also required the app to be written in a language called XCode. However, I quickly learnt that Unity has a feature which allows projects to be easily ported into XCode. I then was able to get everything that I needed installed onto one of the Macs in the Labs in the Queen Mother Building so that I could easily convert my Windows version of the game into an iOS version, and then get it on to my iPhone (which is the primary device that I will be using to test my game).

I also considered the various options for converting the Unity project into various platforms when making my decision because if I do decide to release the app onto the market, then I should be able to expand into other platforms without too much trouble – Unity allows for porting to lots of different platforms, including Android, WebGL, and Xbox.

I also looked into other games which had been created using Unity to see what it was possible to create and how successful Unity-based games can be. I found that there were several games using Unity that had gone on to be successful – two of which stood out for me. The first was a game called “*The Universim*”. This game is a simulation game so I was interested to learn that Simulation/Management games are often made in Unity. The base functionality of *“The Universim”* is the same as what I will be implementing. By this, I mean that the player has different characters (in my case they will be ‘staff’) who can be assigned to do different jobs which then impact on how successful the world (in my case, the Cinema) is.



While this was interesting to learn, I was still not convinced that Unity was the right choice as a lot of the games that I saw had very complex graphics and looked very stylish – which is not suited to my game at all! Then I came across another Unity-based game, called *“The Escapsists”*. This game uses very basic, pixel-art, graphics which were much more similar to my style of graphics. It uses a top-down view similar to the graphics I have produced.

Looking into these two games have convinced me that Unity was the right choice for my game. The fact that both of these games have gone on to be successful is also very encouraging for me as I develop my own game.

The graphics style used in *“The Escapists”* is similar to the graphics style that I am using.

# Graphics and Audio

Graphics

The graphics for my game will be rather simplistic and cartoon-like. This style of graphics is often used for the Management Simulation genre of game, as they are usually just used to represent actions taking place or progress updates. In these types of games, the player does not often have direct control over the objects represented by the graphics (customers, colonists, town members etc.) – they are simply used to show the current state of the Game. As well as this, the graphics look quite fun and light-hearted – which matches the type of game I hope to create.

In my case, there will be graphics to represent the Customers of the Cinema. They will be animated to show how they progress or ‘flow’ through the cinema – from buying a ticket, to going to their screen.

Staff members will also have graphics to represent them. These graphics will be similar to those for the customers although there will be different animations.

As well as graphics for the customers and staff, there will also be some to represent the screens and other purchasable items that the player game buy in the game. These graphics will be repositionable by the player to allow them to set up their cinema in any way that they choose. Again, these graphics will be very simple in accordance with the style of the game.

I will attach animations to several aspects of the game. Again, the animations will be light-hearted and comedic, in fitting with the rest of the game. The main animation I will do is to animate the customers walking through the cinema. This simple walking animation will be needed for the four directions that the customer can travel in (left, right, up, down). The animations will not be excessively complicated, consisting of just 4 frames each. Hence the animation sprite sheet for each customer will be 16 images (4 frames x 4 directions). Customers will also need 2 other animations – one for being idle and one for being in a queue.

Another animation that I will do is making the staff shake around as they get dragged, to make it clear that the player is dragging them. Staff will also need an ‘idle’ animation. If I have time before the end of the project, I will add more animations to the staff to show them carrying out their tasks – to show the player the state of each staff member.

Samples of the spritesheets for these animations are included in Appendix A: Graphics.

I also hope to do animations for other features in the game such as a construction animation for when a screen is being built or upgraded, and one to make the staff dance/celebrate at the end of the day. By including these animations, I hope to further convey the tone of the game, as well as hopefully giving the player something to laugh or smile about, if the animations are funny!

The GUI for my game will take the form of the Cinema. The cinema will be in the same graphics style to the rest of the game – very basic and cartoon-like. Since I am implementing a tile-based floor which is created at runtime (along with all screen objects), the cinema world will appear as almost empty in the Unity editor. The tiles that make up the floor will have a ‘carpet’ texture, which I will make myself. If I see the need to later on in the development process, I will add other floor textures such as marble or wooden flooring. This will allow for users to further customise their cinema to the way they want it.

All of the graphics for this game have been, and will continue to be, created by myself. Since they are simplistic graphics, I am able to use a basic graphics editor to create them – namely, Paint.net. This software allows for basic drawing capabilities which are all that I require for my graphics.

Audio

There will be minimal audio involved in my game. There will be some background music to further convey the tone of the game. This music will be lighthearted and fun – in keeping with the style of the game. This music has been created for me by one of my fellow course mates. I have his permission to use this music in any way I like for this game.

As well as the background music, I also hope to add sound clips to each of the customers. These sound clips will be incomprehensible talking sounds that I hope will help show the success (or lack of success!) of the player’s Cinema. To explain, a busy cinema, i.e. lots of customers, will have a lot of sound coming from it as more customers means more sounds – these sounds will convey to the player that their cinema is busy and successful. On the other hand, if the cinema is less busy, there will be fewer customers and hence fewer sounds.

I hope that, by attaching sounds to the customers, I will be able to convey the success / busyness level of the Cinema.

These ‘Customer Sounds’ will likely be recorded by myself.

I will also add several sound effects for several aspects of the game to provide feedback to the user. These sound effects could include a ‘ka-ching’ for collecting money at the end of the day, a ‘thud’ or ‘success’ sound effect when placing an object etc.

Legal, Social and Ethical Issues

Since I am creating all of my graphics myself, and I have full permission from my course mate who created the music to use it, there are no legal (copyright etc.) issues associated with my graphics or audio. However, I may have issues with a few of the small features I hope to implement. One such feature is the idea for unlocking and displaying movie posters to increase the reputation of the cinema. I plan to do some comedic parodies of existing films (such as ‘Melted’ as a parody of ‘*Frozen’*) but I am unsure of any legal implications that will be involved with these parodies. Since this is a fairly minor feature, I have decided to deal with these issues when it actually comes to implementing the feature – thus allowing me to concentrate on other, more important aspects of the game.

# Code Structure

Object Pool

I decided to use the ‘Object Pool’ design pattern in my game. The purpose of using an Object Pool is to reduce the number of resources used by the game by storing a list of all the available (not in use) resources and then adding them back to the Object Pool once they are no longer in use. By reducing the number of times a new object is created (using the ‘Instantiate’ command in Unity), it frees up processor time and allows for the game to run a lot more smoothly.

I will the Object Pool to store the customer objects. I made this decision because there will be a high turn-over of customers in my game - when one customer is finished visiting, its object will be returned to the pool and can then be used by the next customer to arrive – meaning that there is no need to Instantiate a new object every time a new customer arrives. However, this raises a problem – the Object Pool needs to be set with a size of how many objects to store. However, as my game progresses and the player’s cinema becomes more successful, more and more customers will arrive every day. This will make setting the size of the queue very difficult/impossible. I hope to solve this problem by making the Pool grow-able – i.e. if there is not an object available in the pool, create another one. This means that there will always be an object available when it is needed.

This then raises another problem – if the pool grows in size during the ‘peak time’, then when the cinema becomes quiet again, we are left with too many objects in the pool – so some objects are going unused.

I hope to solve this problem by writing a piece of code that will detect if a customer object has been out of use for an extended period of time, and remove it from the pool if it has.

Update

As well as Object Pool, I will also use the Update Method Design Pattern. This method is run once every frame on every object that the method is attached to. Unity has the Update method built into it which is convenient as it will save some time on the coding side of this project.

Unity will be used in several areas of my game:

* One use of it will be to move the customers across the screen. Every time a frame is executed, the position will be updated and the position of the graphical representation of the customer will be updated on the screen. However, this raises a problem: if certain frames take more or less time to execute than others, then the movement will appear jumpy which is not good! To correct this, I plan to use Unity’s ‘Time.deltaTime’ feature which takes into account the time that was taken to execute the previous frame. This feature results in a much smoother looking gameplay which is ideal for moving the customers across the screen.
* Another use I will have for the Update method is to update the time of day. Again, this will need to run consistently – i.e. differing frame rates should have no effect – so the update method alone will not be enough. I will either use Time.deltaTime again to fix this or I could also use another built in method in Unity called ‘FixedUpdate’. This does almost the same as what Update does but with slightly different running conditions. Instead of running once per frame execution, FixedUpdate will run once X milliseconds where X can be chosen by the developer. This could be useful to update the time because, not only would it be consistent and not effected by the frame rate at which the game runs, but it would allow the developer (i.e. me!) to quickly and easily change the speed at which the day is run.
* The update method is also good for checking for input. I will be using it for this as well – the main use of this for me is for the camera controls. The update method will check for an input from the player – i.e. a ‘pinch’ and ‘scroll’ movement from the touchpad – and then move/zoom the camera accordingly. Since this input detection is not massively dependent on consistent frame execution rates, I plan to just put it inside the Update method – not FixedUpdate.

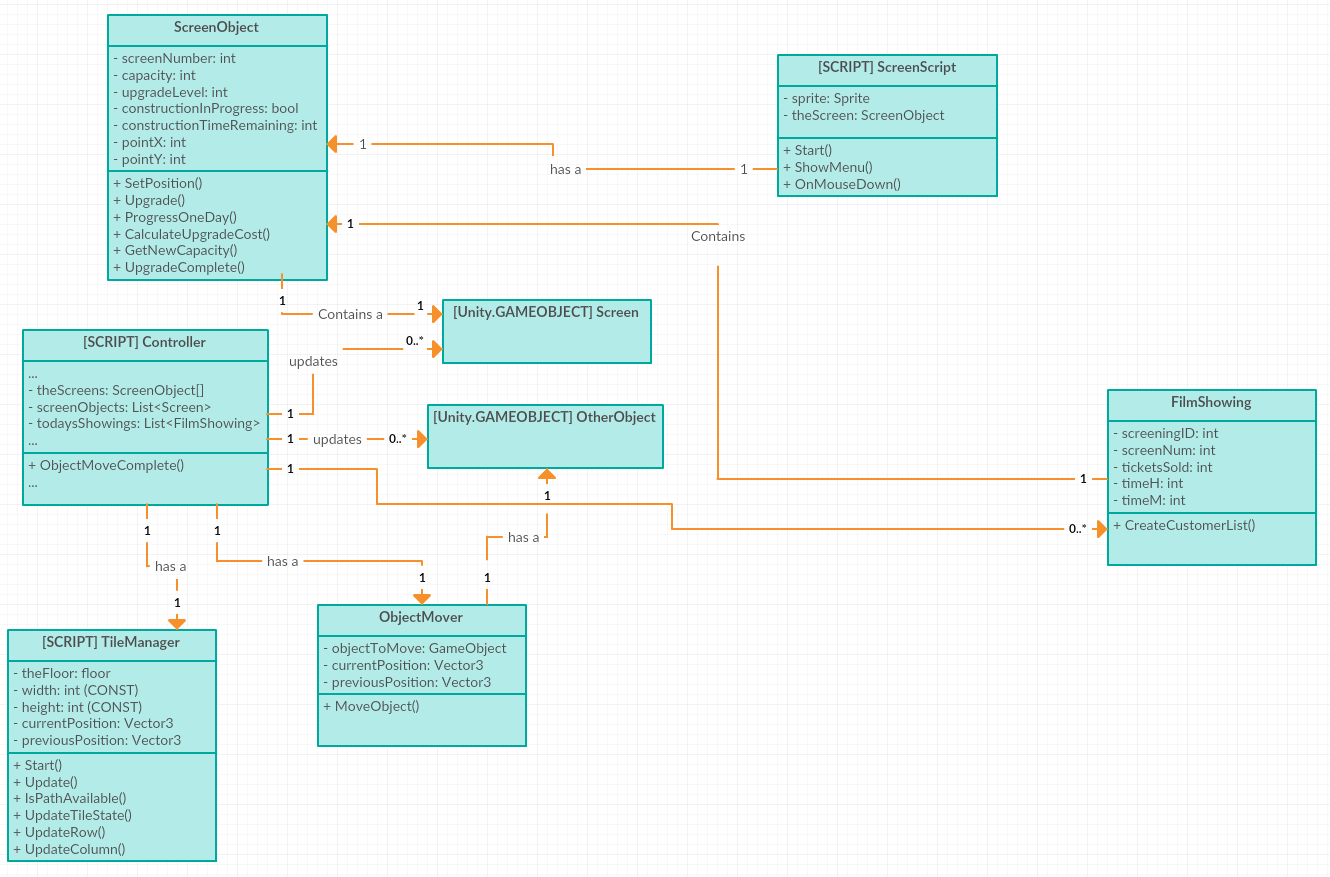
The only concern I have about the Update method is Encapsulation. To fully optimise the Update method, only the code which needs to be in the method, should be in it. i.e. move non-essential code out of the method and carry out its actions elsewhere. This should not cause too many problems but could be fiddly at times.

MVC

I will also use the MVC design pattern in my game. This pattern may prove troublesome to use with the Unity engine since a lot of the front end functionality is carried out automatically. However, I have a rough idea of how to carry it out.

There are two main areas in which this pattern will be used in my game.

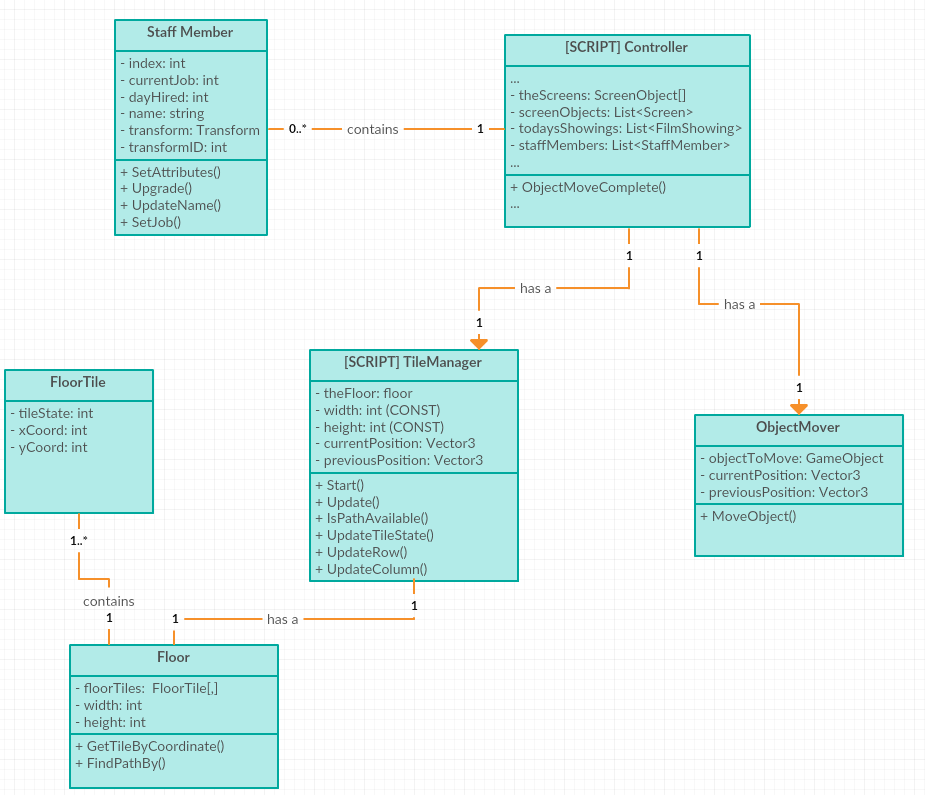
Firstly, I will use it for several tasks that require some back end calculations/updates and also requires these changes to be outputted to the screen. These tasks will include updating the time of day, moving screens and other objects and calculating (and collecting) the money earnt for the day. Each of these features will be coded with a model to carry out the necessary calculations, a view to update the visual representations of the various components (i.e. update a label or move a Sprite), and a controller to link the model to the view.



The class diagram shown here illustrates one instance where I will use the MVC design pattern in my game. The feature in question is moving cinema screens (or other objects such as statues, vending machines, and other decorations) to a new location. This is a feature triggered by the player clicking move, selecting a new location, then clicking confirm. From the diagram, it is clear that the ‘Controller’ class is the Controller in the MVC pattern. This controller will link together the Model which processes of setting the positional values of the object (in the back end) which are used for pathfinding amongst other things, and the View, which will update the graphical representation of the cinema – i.e. show the new position of the object.

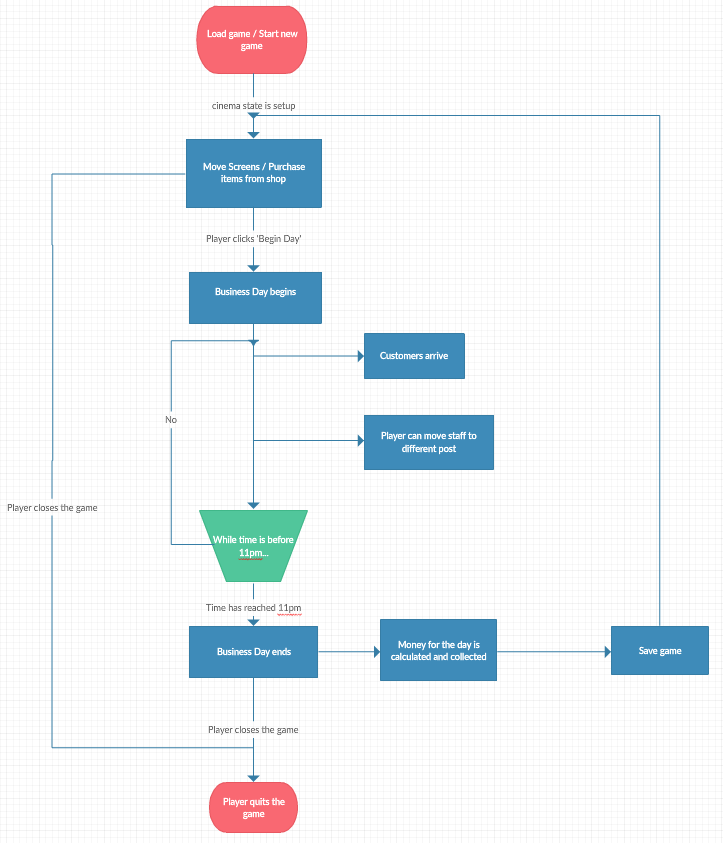
The model of the MVC pattern in this case is the ‘TileManager’ script. It will update the row and column with the right values and also update the state of the affected tiles – signaling whether they are in use or not.

The View for this feature is the ‘ObjectMover’ class. This class will take the actual object (using Unity’s built-in ‘Game Object’ type), and move it to the new position on the screen. Note that this diagram does not include all classes in my game but just a sample of the ones required for the moving object feature.



This second class diagram fills in most of the remaining gaps from the previous diagram. This includes exactly how the Tile Manager will be broken down and how it will work. The script will contain a Floor object, which will in turn me made up of a 2D array of ‘FloorTile’s. This 2D array will

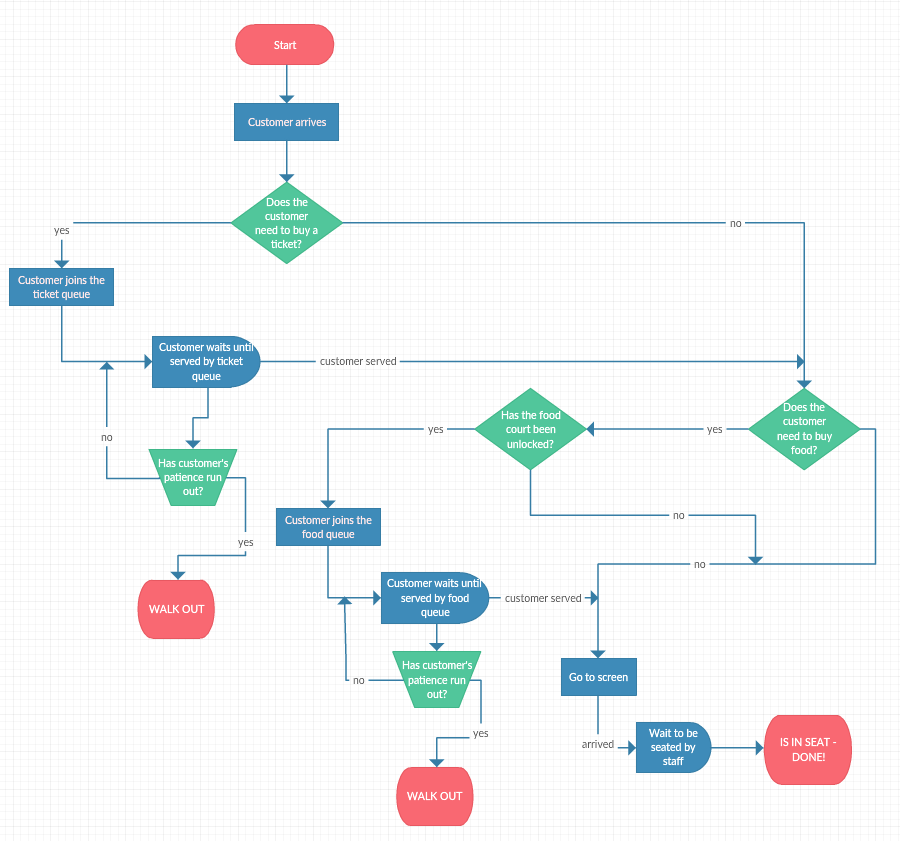
The diagram also shows how the staff members are attached to the controller. The controller will contain a list of all staff members that a player has, and the transform component for the staff member (i.e. the object that will actually be displayed on the screen, is stored inside the StaffMember object.



The flow chart shown here shows the actions that the player takes during the game. Before the ‘business day’ begins, they have the option to reposition the objects in their cinema and also to purchase more items from the shop. Then, once they click the ‘Begin Day’ button, customers will start to arrive and the player’s task then switches to having to manage their staff. The player has to allocate and re-allocate the staff members to the posts where queuing is occurring in order to speed up the queues and optimise the efficiency of their cinema.

Once the time reaches 11 pm, the business day will end and the money earnt for that day will be collected.

My game will implement an auto-save feature that will save the game state at the end of every business day.



This flow chart demonstrates how a customer will progress through the game. At each queue point, the player will have to position staff at the correct ‘post’ to deal with the queue, otherwise no one in the queue will be served and all the customers will all get annoyed and leave!

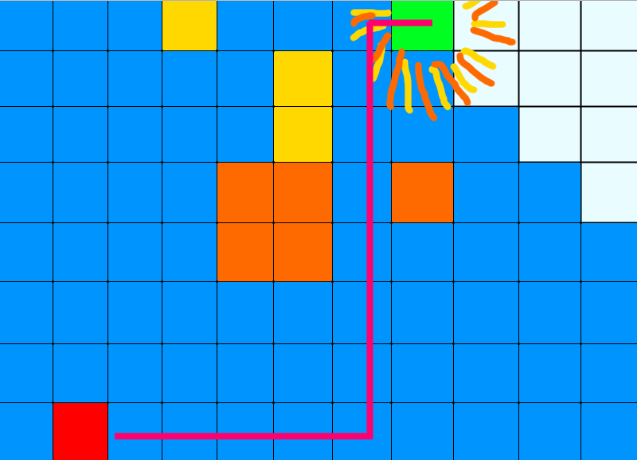
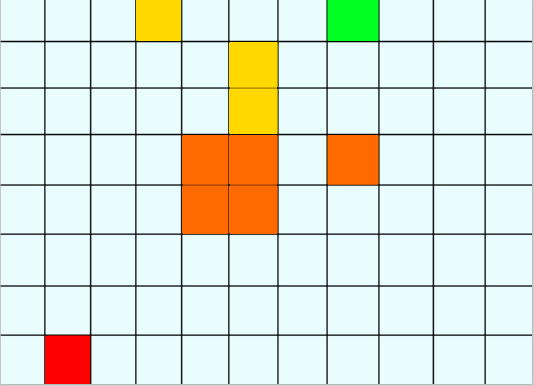
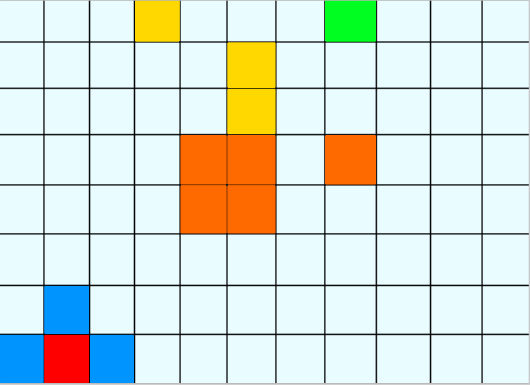
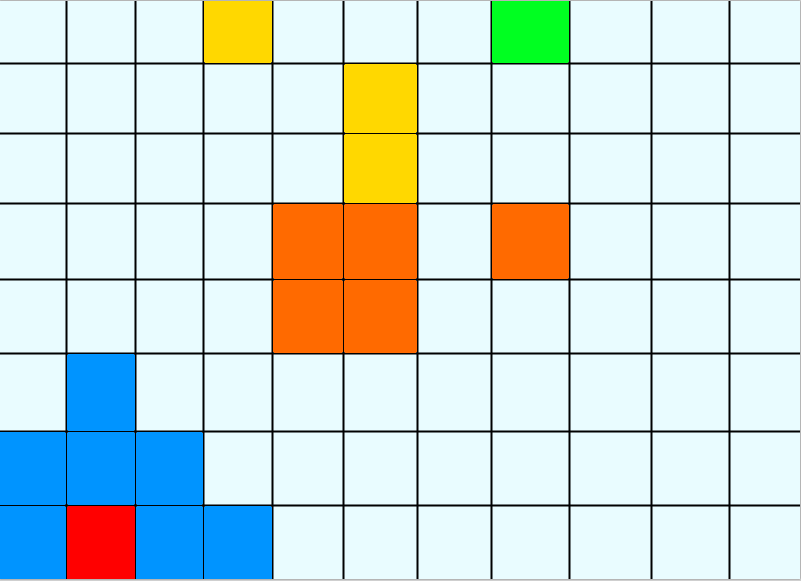
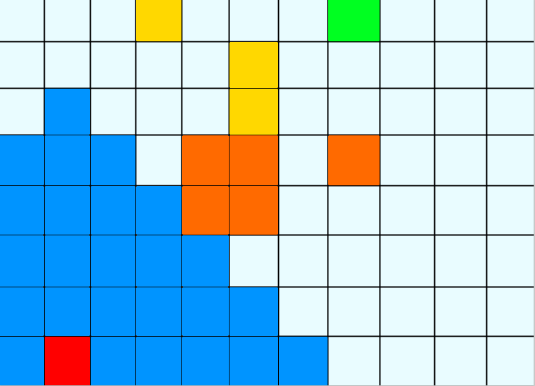
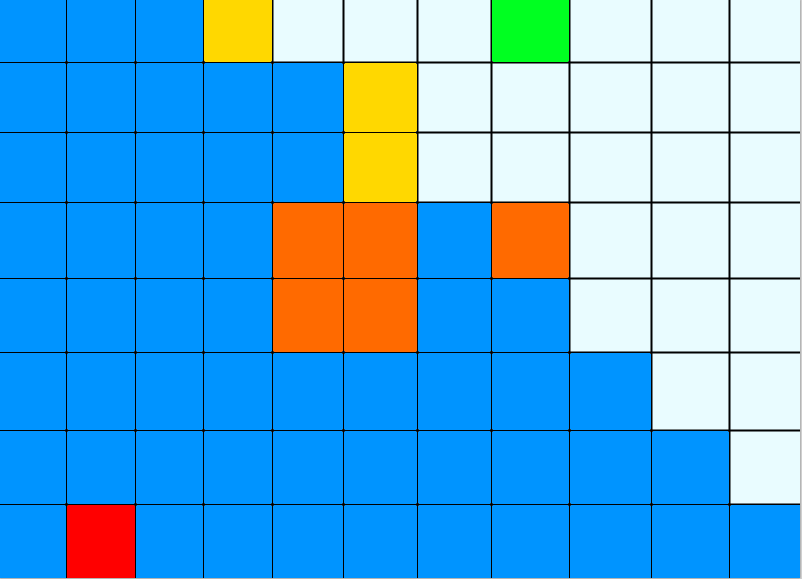
Note that the customers will not all follow the same path – some will require tickets, others will not, some will require food, others will not. This means that the game will vary each time it is run. Some ‘business days’, there will be big queues in some places, other days will result in queues in different places. This will make the game seem more dynamic and realistic represent a cinema – where similar challenges are faced.

# Game Specific Features

My game will include Artificial Intelligence in it. This will be in the form of pathfinding for the customers to navigate a path through all of the obstacles or objects in the cinema to get to the various places that they need to get to (e.g. their selected Screen or the ticket booth etc.). The algorithm I will use for this is similar to the A\* algorithm but slightly different in places. In my algorithm, there is no ‘distance’ to the target stored – it simply stores whether or not a tile on the floor is in use. The algorithm will then move in all directions (up, down, left and right) until it finds the target tile.

My algorithm is not as efficient as A\* but for a limited number of tiles (i.e. the max size of my ‘World’ is 40 by 80 tiles), it is more than efficient enough to serve my needs.

There is also a problem with when to do the pathfinding. If it is done only when the path is required (for example, the customer is done with the ticket booth and now wants to find their screen), this will result in a small delay while a path is found. I plan to counteract this problem by loading all the paths that will be required at the start of a new day, then simply selecting the correct path from the list when it is needed. This is an appropriate method to use as the objects that need to be navigated to/around will be stationary and will not move throughout the day – so the paths will not need to change or be updated.



It can be seen from the diagrams alone that the algorithm I plan to use is not the most efficient! In the diagrams, the red tile represents the start point, and the end point is the green tile. The orange / yellow tiles represent the obstacles in the way – i.e. what to avoid. The blue tiles show the list of tiles which have been explored (each tile is only explored once, so if it has been explored already, it will be ignored). From the diagrams, it is clear that a lot of tiles are explored that would not be explored (certainly not as soon) if I was using the A\* algorithm. If I have time, I may tweak my algorithm so add a distance check into it to optimise efficiency. Although, as I stated earlier, the algorithm is only run at the start of a new day – so efficiency is not a massive problem at that point.

As well as Artificial Intelligence, I plan to have small amounts of concurrency in my game. Concurrency involves running potentially slow code segments in separate threads – thus freeing up the main thread to carry out other tasks.

The main functionality that I will look to split into different threads is the aforementioned pathfinding. Finding a path from one point to another could take several milliseconds – which does not seem like a lot, but in terms of processing it is a very long time and will cause the other processes to wait for it to finish, thus slowing down the entire game. If the pathfinding was moved to a new thread, which could run while the customer waits in a queue, then it would run independently of the main thread which would be left available to carry out other tasks and processing, resulting in the game running quicker/more smoothly – there will be no waiting on other processes to finish.

As well using as multi-threading for path-finding, I also plan to use it for the ‘auto-save’ feature of my game. Since this feature will run at the end of every day, and may take a while to save everything that is needed, I plan to run this in a new thread to allow the player to keep playing the game while the saving takes place.

I also hope to add some online connectivity to my game. More specifically, I’d like to allow the players to be able to log in to, and load saved games from Facebook. This would allow for other features such as players being able to ‘visit’/view their friends’ cinema and compare it to their own. This could lead to competition between friends and more interest in the game as friends compete to make their cinema the most successful in their friend group.

I also hope to add a feature of sending gifts to your Facebook friends. This would be in the form of a mini-game. The premise of the mini game is that the player shakes their device to make popcorn pop, and based on how much popcorn is popped within the time limit, this will be available to be gifted to a friend, in the form of the ‘popcorn’ currency. Again, this may entice more people to play the game more if there is connectivity between friends.

To achieve the Facebook functionality, I will use the official Facebook APK for Unity (a link to which is included in the bibliography). I have not previously used this library so will require to learn how to use it.

# Time Management Plan



The list above shows the basic functional aspects of my game that will need to be implemented.

The longest task on the list is pathfinding which I anticipate being a very long process! I expect this to be quite slow as I will have to perfect the algorithm and the matching code, then make sure it works, then it must be attached to the tile system. This is going to be a long, time-consuming process.

The tasks involving moving and placing object are also expected to be very long. I am not entirely sure how the tiled floor/tile grid will function – I have never coded anything like this before. So I will have to spend some time researching ways to implement this feature.

I also think that some of the Facebook functionality will be time-consuming and arduous. Again, this is a technology that I am unfamiliar with so I will be going into it not knowing anything about it. Hence, it may take several days to learn how the Facebook Unity Software Development Kit (SDK) works, and a further few days to write the actual code.

The hardest part of the Facebook section will likely be the ‘Save game’ part – I am not sure if this is possible with purely the Facebook SDK and Unity (I may need to set up a server and a database on which to store the player details) so, again, I am going to have to research how to do it before actually carrying out the implementation.

However, once I get the hang of using the SDK, other tasks (such as sending invites to friends) should not be too hard – although may require me to read/watch some tutorials about how they can be done.

Changing the appearance of a customer will also take quite a long time. This feature will require a separate menu and a whole new system – there is no similar feature in the game that I can take code snippets from so this will have to be done from scratch. This could be quite a time consuming endeavor, hence I have allocated it several days.

The feature that is the hardest for which to predict a value of how long it will take is probably the ‘Day Functionality’ section. On the face of it, it seems very easy to implement. However, it will involve a lot of experimenting with times for the customers to arrive to leave enough of a gap between them and there will also be experimenting required to ensure that I leave a long enough gap between screenings of films. The income calculation will also likely need to be tweaked as I try to get the values of money right in relation to spending money in the shop.

The ‘other game features’ may end up taking longer than I have estimated in the backlog. As cannot foresee any major problems with these features, hence I left the estimation quite low. However, if I haven’t taken something into account when considering how I intend to implement these features, or if I have assumed that I can do something when in reality I cannot, then this estimation may prove to be too low.

There will be several smaller tasks other than the ones included in the list but the majority of them were small enough that the time taken to complete them will be negligible.

# Bibliography

Game Resources

I have used, and will continue to use, various resources to learn about the various design patterns, code structures and the software that I have decided to use for my game. The main resource I used for learning about the design patterns was the online tutorials created by my classmates for a previous assignment:

* Update Method: <https://teamavocados.wordpress.com/>
* Object Pool: <https://objectpooltutorial.wordpress.com/>

Both of these tutorials were very informative and helpful when deciding on which patterns were best suited to my game. However, the Object Pool one did not contain a tutorial or description as to how the pattern can be implemented, so I found an official Unity tutorial on how to implement such an object pool:

* <https://unity3d.com/learn/tutorials/modules/beginner/live-training-archive/object-pooling>

Although the tutorial is designed for a 3D game, I was able to extract the content I needed from it.

I will be using the official Facebook Software Development Kit (SDK) for Unity which I downloaded from here:

* <https://developers.facebook.com/docs/unity>

When structuring my code, I read part of *‘Game Coding Complete’* by Mike McShaffry to get more of an idea of which game specific features to use. Reading sections from this book was very helpful in my decision to choose AI in particular.

As well as these features, I will also learn by talking through ideas and code snippets with friends and course-mates. This has proven to be useful to me in the past when trying to learn new technologies or new coding ideas.

Report Resources

* My class diagrams and flow charts were created using: <https://creately.com/>
* The charts and diagrams containing statistics regarding iPhone were created using: <http://gs.statcounter.com>
* The image of ‘The Escapists’ was taken from: <http://images.eurogamer.net/2013/articles/1/7/3/0/5/2/8/prison-sandbox-the-escapists-gets-xbox-one-release-date-142116444879.gif>

APPENDIX A - GRAPHICS

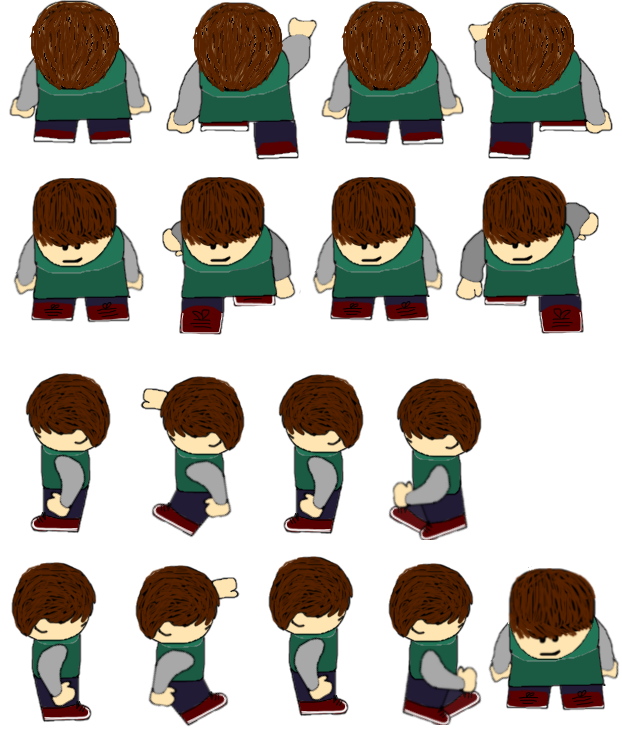
# Staff

* There will be various staff member appearances:
* Each will have a ‘Floating’ animation Sprite sheet for when they are getting dragged by the player:



# Customers

* There will be multiple customer appearances. Each will have a movement Sprite sheet:

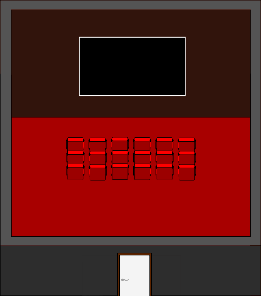


# Screens

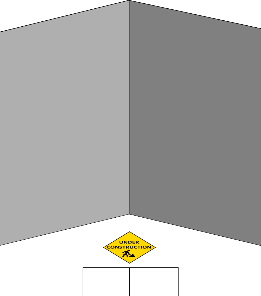
* There will be multiple images for the Screen – for the different upgrade levels:



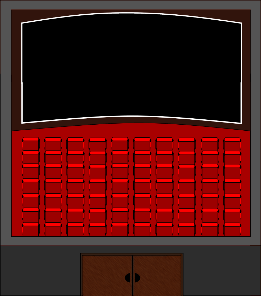
Level 2



Level 1 (Starting Level)



Upgrade in progress



Level 4 (Maximum)



Level 3

# Other Purchasable Items

* Various Different shapes and sizes:



Vending Machine



Plant



Bust/Statue of Game Creator)

# Buttons

* Basic Buttons:



Move Object



Warning



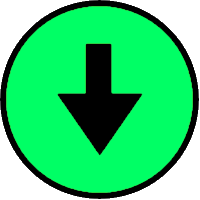
View Info



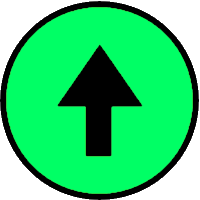
Confirm



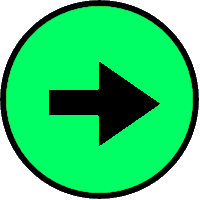
Cancel



Move Down



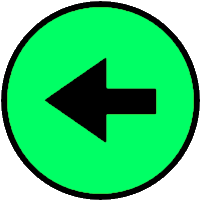
Move Up



Move Right



Upgrade



Move Left

# Currencies

* Two currencies:



Popcorn



Coins

­­­