Revenge of the stick

Final Report

Trello Board: https://trello.com/b/OhD533uO/software-projects
Git repo: https://github.com/akham001/rots

To run on windows click on compile&run.bat otherwise type
"java TestSprite" In local command window

Group M:
Arman, Jethro, Abraham, Mohammed(Nayim Uddin)

Introduction

We as group M created a retro 2d platform shooter called Revenge of The Stick (ROTS). It has a puzzle theme to it similar to older games, but first person shooters have heavily influenced this generation of gamers so we all agreed to incorporate that element into our game. Due to the fact that first person shooters is already such a saturated market we wanted to have a niche, we as a group thought about what we could do and decided that a retro game similar to Aladdin, Prince of Persia, Expendabros and Fancy Pants but to also include a shooter element into the game as well.

Our game revolves around a character fuelled by revenge due to the events that have taken place, which has cost him everything, he has nothing left to lose, therefore all he wants to do is seek revenge. He navigates through dangerous environments being confronted by foes that are in his way.

The game is was designed with the aim to create a very relaxed but still engaging enough to immerse the player into the game so they can enjoy the game and become invested into enough to make the player want to carry on playing.

The gameplay consists of puzzles and platforms with very simple and blocky movement and gameplay however it will include numerous enemies to kill, it will require a little bit of attention from the player but they will be able to sit back and play it after a long day at work/university to unwind.

We aimed to attract the older gamers as our main user base however we wanted to Make sure that we would create something that everyone would enjoy so that our user base can expand
As a group we talked about how the game should work and what our favourite games had in them to include into our game as well. These included things such as how the camera would follow the player around, what each stage looks like, should it be a room-to-room game like a dungeon crawler or should it be one long level like Mario or Sonic the hedgehog.

In this report we will outline our development process, how we came about and discussed all the ideas everyone presented. As seen in our proposal we sent out a survey monkey to get feedback on our ideas and what was good or not. We found out that the users wanted quality gameplay over quantity. Furthermore we covered topics such as open world-ness, difficulty, storyline and graphics, and what they rank in order of importance to the player.

We will also discuss how we worked together, which version control programmes we used, how each of them helped, how different they are and how they helped in different ways. It will cover our meetings with each other and our supervisor Nikolai, how he guided us and what we learnt from him. We will give an overview of our product, our implementation, and what we changed from our initial proposal.

Development Post-Mortem

We initially set out like last term and assigned each person in the group their own personal piece of work that they had to do, this was decided upon in regards to the strengths and weaknesses of each member of the team.

Arman and Jethro both worked on the backend of the game and the functionality such as the game engine, the implementation and algorithms. Abraham worked on animations, sprites, the images, buttons and background. Mohammed worked on research, uml class diagrams, documentation and planning.

Our 5th member Nikhil has not done any work, has not replied to us and has contributed nothing towards this term at all.

We met on a weekly basis, to discuss the ideas and issues we were all having separately on the pieces of work we were working on. Our first meeting consisted of getting a way to all communicate together, we agreed on WhatsApp and created a group chat so we could all communicate easily. We discussed which programming language we would use to create the programme and we agreed on java. This was because we all had experience with java. We as a group talked about what kind of game we wanted to create, after talking about multiple games that we liked, we agreed upon a very simple but engaging game. We then set about creating paper prototypes to see what our game would actually look like so we can make improvements on our ideas before we code it. In our appendices we have attached the paper prototypes that we made.

Our meetings with Nikolai were very helpful, he pushed us to create the uml diagrams and to plan everything, give everyone their own piece of work. His guidance helped a lot with timekeeping and team management. Further along into the term he told us to create functions, and what they would do in classes on their own which Mohammed made with the help of Jethro. This made the idea come alive in our heads and we could see what needed to be done and how to implement the code together. This way of working made our code modularised and very easy to connect together to create a working product.

Overall our group worked very well together apart from the last member Nikhil who did nothing and did not try to communicate with us. If any of the group members had problems we would solve them very easily for example the button images that Abraham created were not transparent so all it took was a quick message in the WhatsApp group and he provided images with transparent backgrounds.

We as a group used multiple version control programmes, first off, everyone in the group had a trello board of their own and every other member was given admin permissions on it, the group member would then use his trello board to upload any work that they had done, along with ideas and suggestions for what could be improved. The actual code was uploaded using git so Arman and Jethro could work on it together, the link to the git repo is https://github.com/akham001/rots. The use of

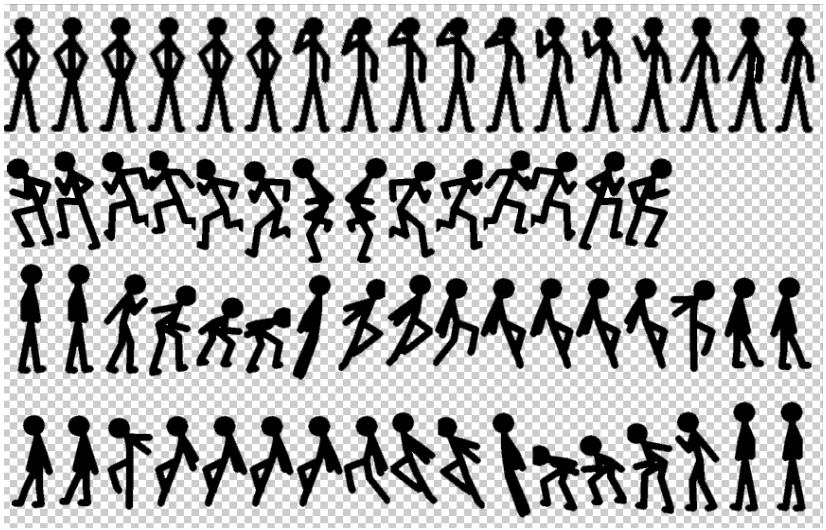
git helped a lot as we could quickly edit and upload the code without coming into conflict with each other and if we did it could be easily resolved.

The presentation that we all worked on and presented together was very successful and went exactly as planned. That was one of the points where our planning and group worked together very well. Each member of our group apart from Nikhil had their own part, we created the presentation together during one of our meetings so everyone knew what was going on in case someone forgot to say something in the presentation. Every member had to do 2 slides each and present the work that they have been working on, it went very smoothly we answered every question to a high standard, Jethro gave examples on modularised our code is that it would easily be ported to phones.

Due to us meeting as a group once every week, sometimes more, it sped up our implementation process as we discussed everything in detail before we coded anything. We discussed enemy ideas, such as movement patterns, health, AI, how they interact with the play, this can be seen in the appendix (1,2). We also discussed what our stages will look like, how the player goes from stage to stage, where the camera will be, what sort of movement the player will have.

An issue we ran into was how to create the sprites and how they would be loaded in. All the animations created was saved as separate images, for each frame the stickman was positioned in different places, there was too much white space around the each frame of the stickman and the stickman wasn't centered to the middle of the animation sheet. This was a problem for Jethro and Arman because they set a specific parameter for the stickman to be in the sprite class. When we loaded the separate images we discovered that the frames were out of position and overlapping each other.

We resolved this problem using Jethro's program which gathers all the images from a folder and puts them all in a row. We ended up creating a contact sheet which had all the stickman animations which easier to load in the game.



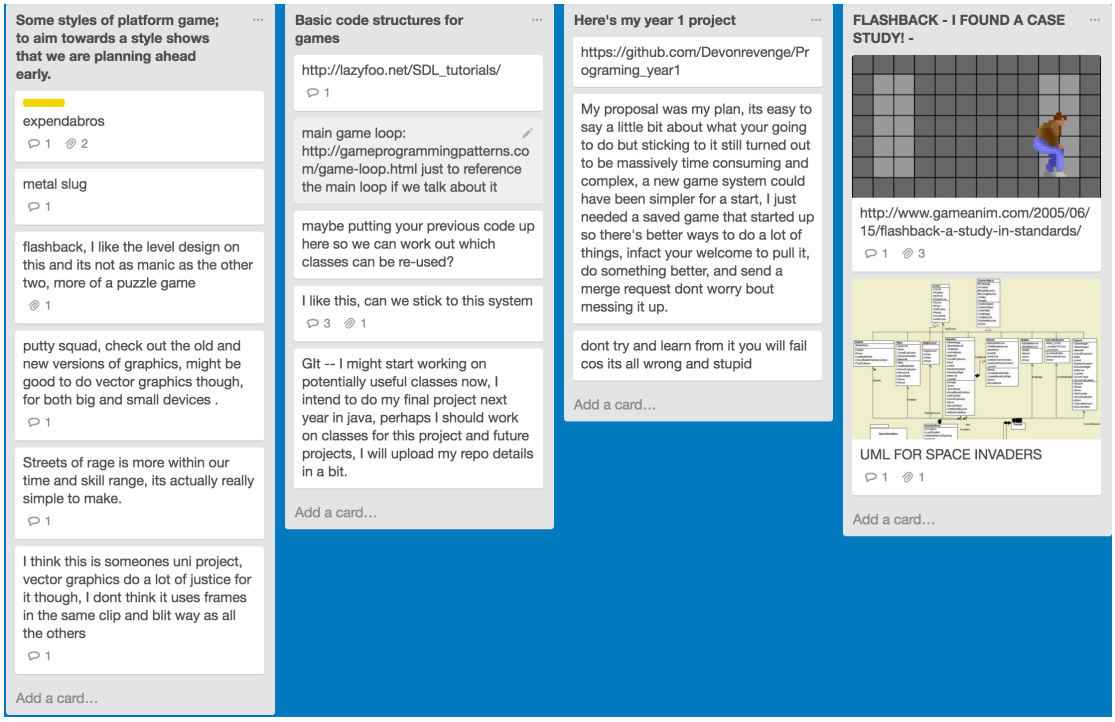
Formative Evaluation

There were many evaluative aspects we thought up as a group after we created our first working version of the game. We all agreed to name it ROTS version1.0. The aspects we focused on our analysis of the game were objective. They were to see if there are any errors in the game. If the game stopped at any point and crashed. We also checked for glitches in the game if all the platforms were solid and also if all the pick-ups such as bullets, health and guns worked effortlessly. In addition, we checked if the enemies died when being shot at and if the health meter worked on harder enemies, as they would need to be shot at a few times before they died. As well as this, we checked if the game saved and if the score was saved to the high score.

Each member of the group tested each of these functions of the game so the results were less biased. They were tested on different platforms and operating systems, some of which were Windows operating systems and IOS. The game worked very well on both platforms without any issues objectively, but we quickly realised that we will nevertheless, have biased views towards it as we built the game ourselves and we will not analyse and evaluate the game properly. We met up and discussed this thoroughly and came to a decision that we needed external research done before we could analyse and evaluate our game properly.

We used software applications such as WhatsApp, Trello and Skype to communicate frequently. We also had face-to-face meetings as a whole group to come up with ideas to test the game in various ways. After much debate, we came to a conclusion that user experience and usability were of utmost importance. We realised we can get external research data and analyse through the data gained, to improve our game to a better standard.

This is just an example of one of our boards on trello



Objectively the game was very good and on track but we had no subjective data to improve the game. We needed external participants to play the game and give their individual view of their feeling and opinions towards the game. This included user experience and usability and we quickly realised that we needed a good number of people to play our game, who wouldn't have any bias towards our game.

We set out around the university as a group and asked students if they were busy and if they could spare 5-10 minutes of their time. After an hour of searching and talking to students, we found 11 willing students who were the number of users, to play our game and tell us how they felt about it after. They were told that they will be able to play for 5 minutes and then will be asked questions regarding their experience. After that, they will be able to play the game for a further 5 minutes, after which they will be asked a few more questions. They agreed and we took the group to one of the labs in the university.

We loaded up the game on different machines and operating systems, ready for the 11 participants we found through means of casual conversation. They were not told anything about the questions they will be asked about, after they played. They were only told about the aim of the game, the backstory of the game so they feel a need to want to play and also were told about the buttons which are used to play the game.

After they played the game for 5 minutes, we told them to turn their attention to us and we started asking the participants questions. The questions which we asked them were based on their experience of the game and the usability of the game too. They were asked to answer subjectively, which some of them didn't understand, we told them to answer the questions with their feelings towards the game and also their opinions too. They understood this and the interviews of the participants started thereafter.

The questions they were asked related to the experience that the users felt after playing our game. These included effectiveness, efficiency and overall satisfaction for

the user. This also referred to the user's experience and how they felt about the game. The questions that they were asked were very broad, allowing long subjective answers, which would allow us, as a group to gain extensive research to improve our game. One of the questions related to the ease of learning the controls and what to do in the game to complete it. Most of the candidates found the controls very simple and easy to follow. We also have a user guide which we created to allow users to understand how to play and what button does what. They all found the game to be very simple to play as well as finding the story very enticing. They wanted to play on to carry on with the story and find out what happens next.

All the participants were asked questions regarding the design and if it was effortless to understand how the game works and how to navigate through the game. As well as this, they were asked if the functions of the game were useful and if they fit the purpose. In addition, we asked them if the game was easy and if it was efficient to get things done. Furthermore, they were asked if it had attractive aesthetics, if the game was visually attractive and if the game gave them a blast-from-the-past kind of experience and if it gave them inspiration or wowed them in any way and why. Finally we asked them if there were any errors or glitches while they played and if every function worked accordingly.

After they were interviewed, we wrote down what they had told us and we asked them to play the game again. The second time we let the same participants play, we didn't tell them the rules of the game or what each button did. We wanted to see if the game was memorable and if they picked up on those aspects quickly or not. This also showed us the efficiency of how fast an experienced user can accomplish tasks in the game.

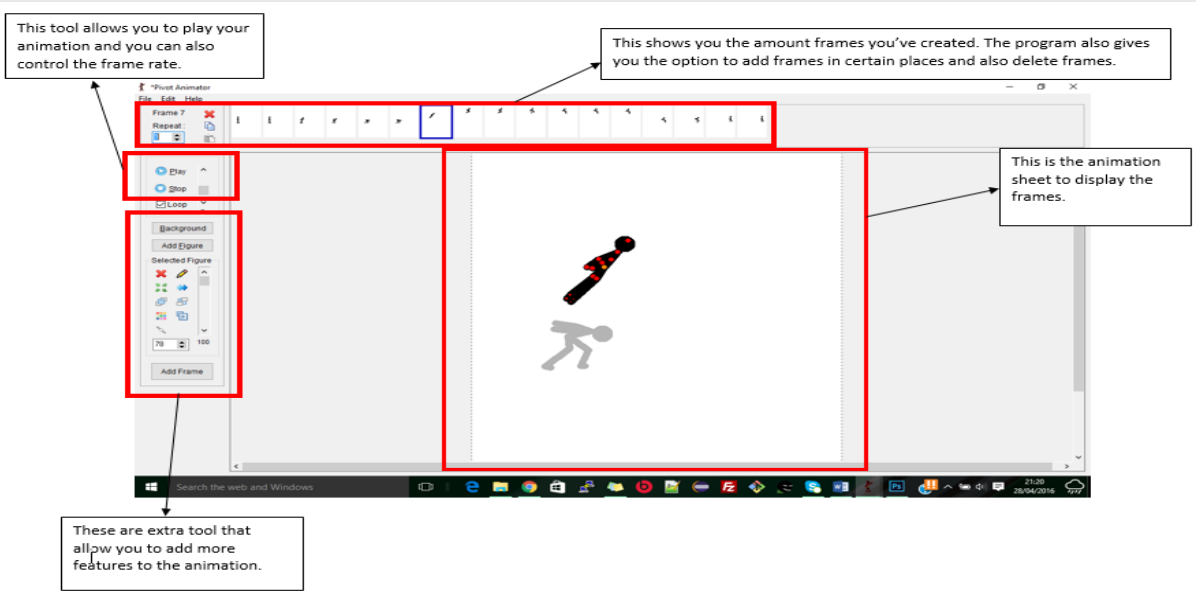
Only one version of the game was tested, thus far, which was ROTS version1.0. However we took all the participant's emails down and asked them to play our improved version of the game which would be emailed to them. They all agreed which was very helpful. We improved our game which mainly depended on the data we collected from the participants beforehand. We named the new and improved version to ROTS version 2.0 and emailed the participants to play the attached game, and also fill out the attached questionnaire which had similar questions as to when they were interviewed.

Design and Implementation

As a group we started off drawing a paper prototype of our game, we chose this method because we thought it would be a quick to get our ideas together and add any extra features and improve our initial design. Creating a paper prototype gave us a mind-set of how the user would play the game it allowed us to explore any small details that could be added to our initial game to make it better.

All the artwork was created using various applications we used Photoshop 2015, Gimp, Paint and Pivot Animator. Abraham used Pivot Animator to create all the stick animations frame by frame. There were different types of animations created such as running and jumping and an animation that allows the stickman to jump on ledges and bring himself up to platforms. There were tutorials available on YouTube that demonstrated different techniques on how to create these types of animations we used

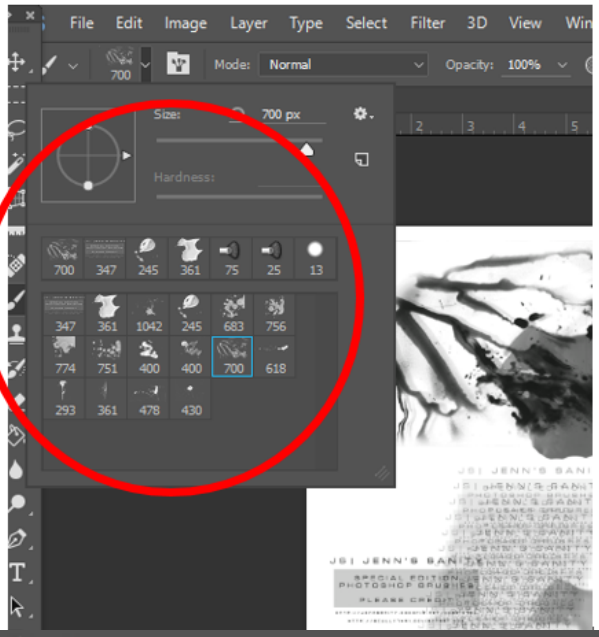
this as a guide to create our own style suitable for our game. Once we create a piece of animation in Pivot animator gave us the option to save each frame as separate images. Each type of animation was separated into folders and was uploaded on trello. Abraham uploaded all the animations and artwork on trello for everyone to see and saved each type of animation as GIF files so that everyone can get visual view of what the animation does.

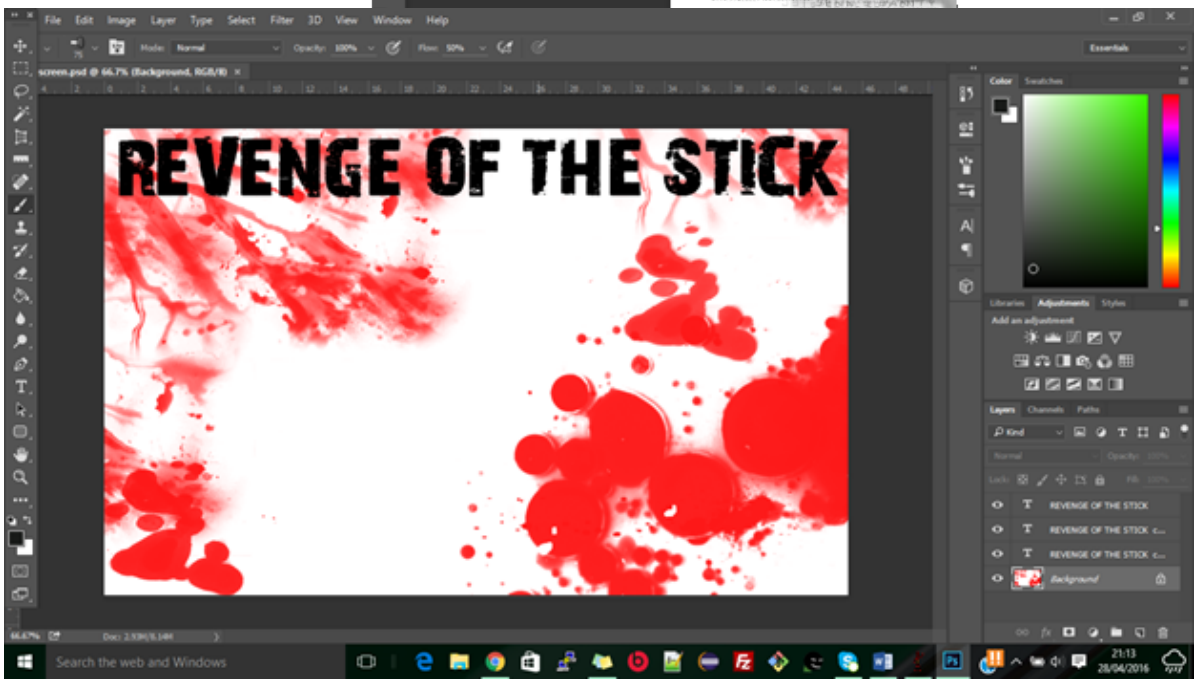


Here is a guide of how Pivot animator works:
All the other artwork such as backgrounds and buttons was created using Photoshop. As a group we thought about how our home screen should look like and the sort of colour scheme that would suit the storyline of the game - the stickman seeking revenge against creatures that attacked his hometown. We decided to create a gruesome home screen to indicate users that the game will involve violence. Here is a screenshot below of the home screen. As you can see below, Photoshop was used to create it. Abraham used a tutorial as guideline to create his own version of the background. The brush tool was used to paint out the blood splatter, the brush was downloaded then imported into Photoshop ready to use (http://scully7491.deviantart.com/art/Blood-Brushes-III-20634082). We checked if the brushes were for non-commercial use only and it was along-side with the font (http://www.dafont.com/28-days-later.font).

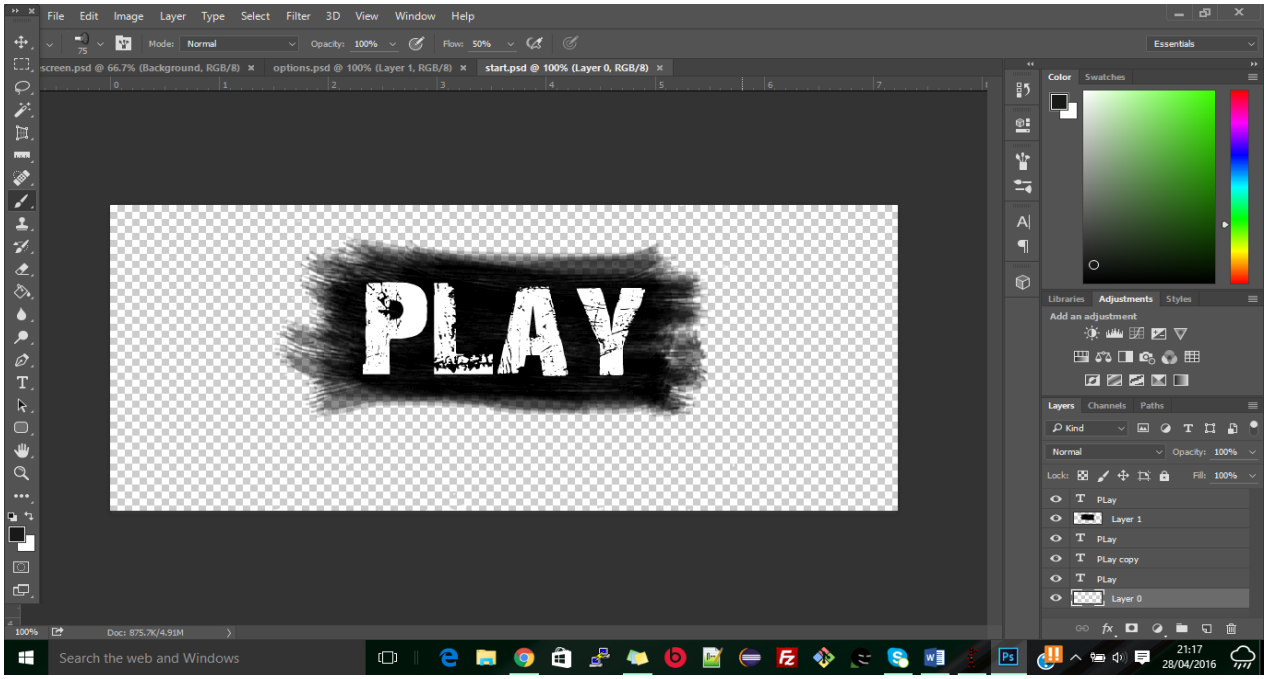
Here is a screenshot below of the blood brushes that was used to create the home screen background, as you can see there were different types of brushes to choose from to create whatever you want.

After getting comfortable with the using brushes Abraham was able to create the background for the home screen. Here is a screenshot below:



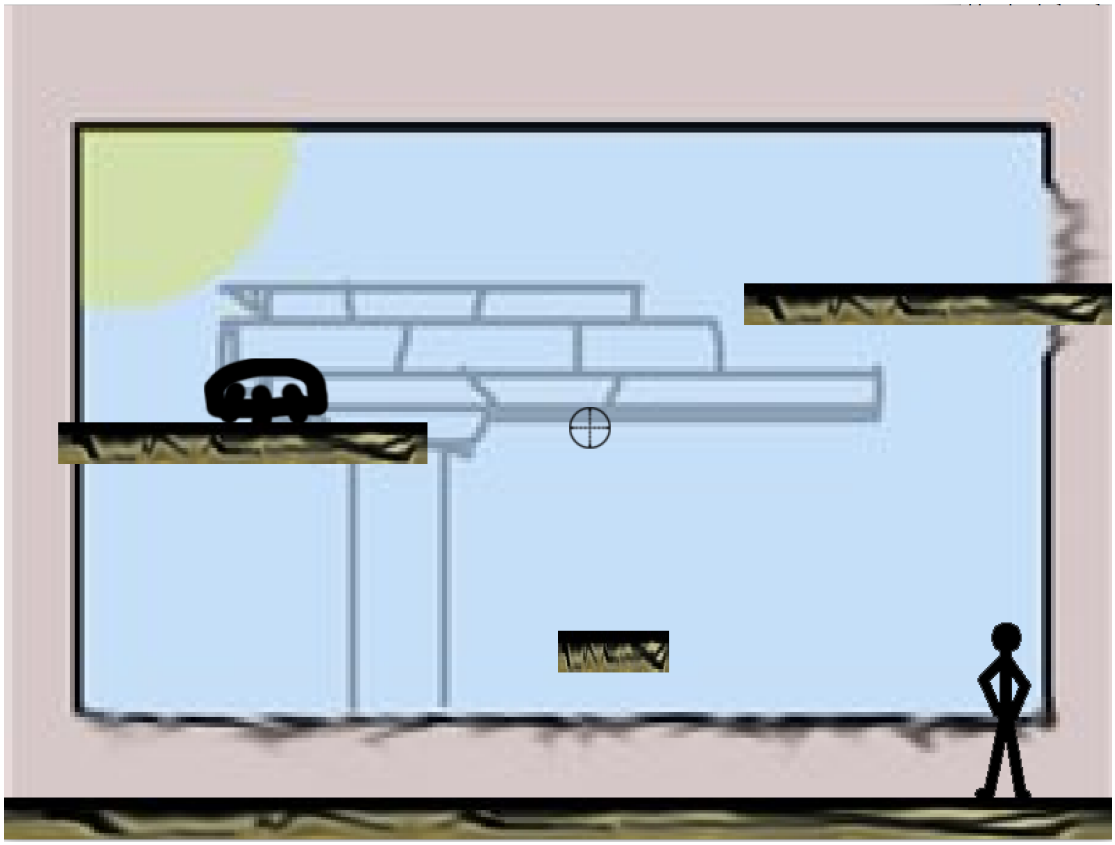


We thought about what type of buttons we should design, we started using simple squared buttons and then Abraham thought since we used blood brushes for the home screen background we should maintain the same idea and use a simple brush in Photoshop and to design the button. Here is another screenshot below to show how the button looks like.



Here is a screenshot of the home screen:

Here is a screenshot of the gameplay:



Java was considered appropriate for the game as processing was familiar to us and it is similar but faster and it also has a graphics API that works the same on all systems, this suited us for development reasons as well as the market we aimed the game at.

As the game utilises a lot of sprites, as in buttons, platforms, baddies, player, bullets and a lot of other functionality planned for later stages of development, It was very clear a ‘Sprite’ class was needed, a sprite in this case is an object which draws an image or animation relative to its dimensions and what it is doing.

Managing the sprites behaviour depending on what it was supposed to be doing became an issue, because the sprites are needed to do so many different things a lot of functionality needed to be built in such as collisions, particle emitters, abstract strategies for pre-programming automatic movement and movement based on key presses the sprite class became big and complex to use, although the class was written with a lot of comments so the user could learn quickly documentation on how to use the sprite class had to be produced so members of the group knew how to use it.

A guide was made ‘Using the Sprite class’ and is included as a deliverable for the project.

Another consideration to make a complex class easier to use was to include in depth error reporting and to force sprites to be initialised in try catch blocks by having the class throw errors with messages, with plenty of comments included users should be able to find the error faster.

The sprite class images were initially meant to be drawn to Java's Jpanel however the many functions and operations would not suit drawing to a buffer first so research led to using a class that extends Canvas and drawing that to Jpanel from within the function using a 'double buffer strategy' instead, this suited the program well as this class also supports mouse and key presses as well as implementation to run the main game function in a separate thread in Jpanel.

The implementation of the project was developed with consideration for the future, that is to say that the sprite class is designed to be used in a modular way and a sprite object can be broken down into data, for example, by unloading each of its arrays containing image, position, angle, speed, etc onto a text file, another object or objects could be initialised with this data simply by populating the right arrays again, this could suit saving the game or even online multiplayer functionality, as only low level integers and strings need to be passed over a network or onto a smaller file in order to represent a lot of complex workings.

Quality Assurance

To assure the quality of our game we had many ways of testing it to get it up to scratch. Objectively, we ourselves, tested the game on many different systems and played it many times to see if there were any bugs in the game or any glitches for that matter. Individually, we sussed out certain issues we had such as the player not being able to jump on to the platform. By playing the game a number of times, most of the glitches and issues were found and changed to make the game run smoother. Most of the issues we ran into, we found solutions for quite quickly because we communicated with each other effortlessly through many means.

The results which we gained through our extensive research was very valuable at each given moment. This is because the analysis and evaluation helped us improve our game somewhat very quickly. The results were very subjective as that is what we wanted, this is because it is a game and the user's experience and feelings towards it, counted a lot towards if the game was a successful or not. This is because a game is supposed to be entertaining and immersive as that is the main purpose of any game, to entertain users and to make them want to carry on playing.

There were many ways in which we had question and answer sessions to gain unbiased views towards the game. They were acquired through face-to-face interviews and also questionnaires which were given out to 11 participants. They were asked to play the game, then answer a few questions and then play again. After playing the game the second time, they were told that they would be sent an improved version of the game through an email as well as a questionnaire which they were

asked to fill out. The results we got back from these sessions were very helpful to improve our game.

The results we got from the first time we asked the participant's questions after playing the game were very detailed and in depth. The results were not as expected for the ROTS version1.0. The users found that the game was not very smooth in terms of the player's movement and their actions such as jumping onto a platform/ledge. As well as this they did not find the aesthetics of the game to be very enticing or visually attractive. They also said that the game didn't stimulate them or give inspiration to them. In addition, the users did not find the game important and did not subjectively, like the game as much as we hoped they would as they told us that there were no sounds in the game.

However, after the second time they played the game, and they were asked questions again, most of them found the controls of the game to be very easy to learn, the functions fit the purpose very well and were memorable. They also found that the design was intuitive and understood what the aim of the game was without any effort. None of the users found any errors while they played which was good as all the pick-ups and guns worked efficiently. As well as this, the usability and efficiency of the game was very good as most users found it was easy to get things done as they understood what to do and we also realised that when they played the second time, they accomplished tasks faster than the first as they got used to the controls.

We were disheartened by the results we gained as there were more negative feedback than positive. However we did not let that dishearten our morale and we quickly met up as a group and discussed the results we had gained from the Q&A. We jotted down what were the main issues of our game and delegated parts to each other to complete. Abraham was given the task of creating more sprite images to the game to make the player move smoothly as the more frames there are, the better the transition will look. The player will walk and it won't look like it is skipping frames. He was also asked to improve the background of the game to increase the aesthetics of the game. Jethro was asked to add more enemies and platforms to make the game a little harder and Arman was asked to find sounds for the guns, enemies and when something is picked up in the game. These were the improvements we set out to accomplish so users don't get bored quickly and it stimulates them enough to find the game important.

After we improved the game to a better standard, we named the improved version of the game to ROTS version2.0 and were happy with the result. We had added every aspect we set out to after our first user testing that we did. We attached a questionnaire and the game to an email and sent it to the 11 participants to play again. In the email, they were asked to fill out the questionnaire and send it back to us after they played the improved version of the game. After they played the second version of the game, the results we got back from them through the questionnaire were much better than that of the first version of the game.

The results showed us that they found the design to be more intuitive and understood how to navigate through the main menu as well as the game effortlessly. As well as this, the aesthetics of the game, improved which stimulated the users and they liked the game more than the first time they had played it. This showed us that users found the game more important and also subjectively liked the game much more in the second version than the first.

Nevertheless, they did not like the fact that there were not enough sounds to immerse the player into the game. There were no sounds when the player gets on a ledge. They told us that the second version of the game was a vast improvement, but there were still issues.

The final game conforms to most of the initial requirements that we had. It does run on multiple system, the user experience and users usability were up to a good standard.

Due to the the nature of our group, some members of the group have not actually worked on the coding part of the game so we used this to our advantage because when we implemented a certain aspect of the game such as gravity or jumping we would show them the game and tell them to play it, we would not tell them what it was supposed to do this way they were subjective in their testing and would find any bugs. This was very effective because users normally are not that thorough with their testing but because Abraham and Mohammed were in our group and they wanted to iron out bugs they tried their best to find any. This led to multiple testing sessions with both of them not being told what to look for but told what we implemented.

1.) First Test - gravity & platforms:

We implemented a gravity element into the game which would work on all sprites that have their gravity set to true, they are pushed down, however this angle is also changeable. This means the player is always being pushed down until it collides with something, in this case we also implemented platforms into our game.

The expected outcome was that the player would land on the platform once he had spawned which would set collision to true and disable gravity so he lands perfectly on top of the platform

The outcome that we found was that the player sprite would land inside the platform, we debugged this and realised what was causing the issue, due to the player falling at a certain speed it was skipping past the top of the platform but then colliding the with middle and stopping there, therefore we created a function that would push the player to the top of the platform if he went past it.

2.) Second test - jumping & running:

We implemented jumping and running into our game which would work with the sprites Abraham created, the speed of running and jumping, the gravity and acceleration were all controlled with the functions which gave us the opportunity to do anything we wanted to as we could edit how high or what angle the sprite jumped at or how fast he would run.

The expected outcome was that the player would run at the speed set and jump at the angle that we wanted to to make the sprites look very fluid and to make it run and jump at the angle that the sprite was created for

The outcome was very positive and we created a very fluid movement in terms of running and jumping, it looked realistic yet blocky and retro which was exactly what we wanted to achieve.

3.) Third test - collisions & crab:

We implemented an enemy for the player to fight against, a little crab that walks along its own platform, however we would have also needed to create collision between that and the player to create any sort of interactivity therefore we implemented collision between the crab and the player.

The expected outcome was that the player would collide with the crab and it would execute the logic that we wanted such as the player losing health or being knocked back.

The outcome was as expected, there were a few glitches where the crab and the player would detect collisions outside of their sprite as the actual sprite is a box and would look like no collision should take place but it was, therefore we had to change the collision radius of the player and the crab to match what was being shown on the screen.

4.) Fourth test - shooting and crosshair:

We decided the incorporate a shooting element into the game because our market researched showed that a lot of users enjoy shooting and just general violence, this lead us to implementing a bleed effect for the crab when it was shot. We came to an agreement that we would create a crosshair that the player would control with their mouse which allowed them to shoot in any angle they wanted to, due to the nature of the game being hard and the player only being 1 hit from death meant that the player needed to be able to kill the crab without getting too close to it.

The expected outcome was that the crosshair would be positioned on the mouse, the bullets would shoot in the direction of the crosshair however we also added a random element into the shooting which meant that the bullet would shoot with a little random angle added to it to make the game harder and more interesting and not repetitive. When the bullet would collide with the crab it should make him bleed and die.

The outcome was as expected, the crosshair was in place of the mouse, the bullets shot towards the crosshair however it did not have 100% accuracy, once they collided with the crab it would make him bleed and die.

Summative Evaluation

We released our product amongst our peers and friends to get a general feeling of the public interacting with our programme. This meant that we got to test it out on multiple different systems with different specifications.

The feedback we got from the users was very similar to the feedback we received before. The general consensus was that the game was very engaging to play and the general feel of the game was very fun. Due to the game being retro in feel, this was achieved by the blocky movement of the user and the simple yet effective AI of the enemies. The game is very simple in its core but it can be expanded to become something big. Most of the users agreed with our statement that the game was very easy to play but the user could immerse themselves into the game and want to carry on playing.

The negative feedback was similar to the concerns that we had within our group ourselves, everyone wanted to better and better their own piece of work and thought of extra features to add on. Some of these were processes that required a lot of time something we did not have. One of these ways was to have extra loot and items dropped from monsters, these could range from items such as ammo or guns to pickups like shields, extra health or armour.

Another improvement to make was more fluid and responsive animations, this could have been achieved in multiple ways such as more detailed frames, with additions to items of on the sprite such as guns or clothes that also are responsive to each frame creating a much more fluid feeling while watching and playing the game. However due to time constraints we could not achieve this desired result.

Furthermore the users suggested that sounds effects for certain things such as gunshots, certain animations such as jumping and climbing should also have its own sounds. A distinct sound track to match the game itself to make it more immersive, and memorable.

Multiplayer functionality was also mentioned multiple times by our own team and a few of the final batch of testers. This would have been made very easily implemented due to the nature of the code and how modularised it is. However due to time constraints this was not possible. All that would have been needed was another sprite, and 1 class that contains the new player and all the variables for it, such as the sprite, gravity settings, health etc…

Overall we wanted aimed to achieve a game that had a retro feel with an easy skill cap but engaging atmosphere and we have achieved this. The game looks very retro with the graphics and sprites that don't look like their from this generation. The blocky movement was followed by the fluid movement of the sprites. It was a very fun experience working as a group, as we all learnt new ways to achieve what we wanted for example learning new programs like gimp, photoshop, pivot animator etc.

In conclusion we have managed to stay on our guidelines and achieve what we wanted to achieve, albeit on a subpar level to what we wanted it to be, the lessons learnt from this will push us to work harder and more efficiently next time a project like this is encountered. Team management skills such as time management, distribution of work, dealing with different personalities and working on the same programme with multiple people have all been new skills that we have learnt and got better at as a group.

Bibliography

A list of published sources referenced in the proposal.

Blood brushes used in photoshop:
http://scully7491.deviantart.com/art/Blood-Brushes-III-20634082 - This is the link to deviant art website and the creator of the brushes and it has non-commercial licence.

Font:
http://www.dafont.com/28-days-later.font - This is the link where we got the font used for the homescreen and throughout the game. The font was free for personal use.

This link covers the basic implementation used for sprites:
http://lazyfoo.net/SDL_tutorials/

The following two links explained how to go about setting up an environment that suited sprite objects:

http://content.gpwiki.org/index.php/Java:Tutorials:Double_Buffering
http://www.java-forums.org/new-java/51185-copied-code-tutorial-bufferstrategy.html

Appendices

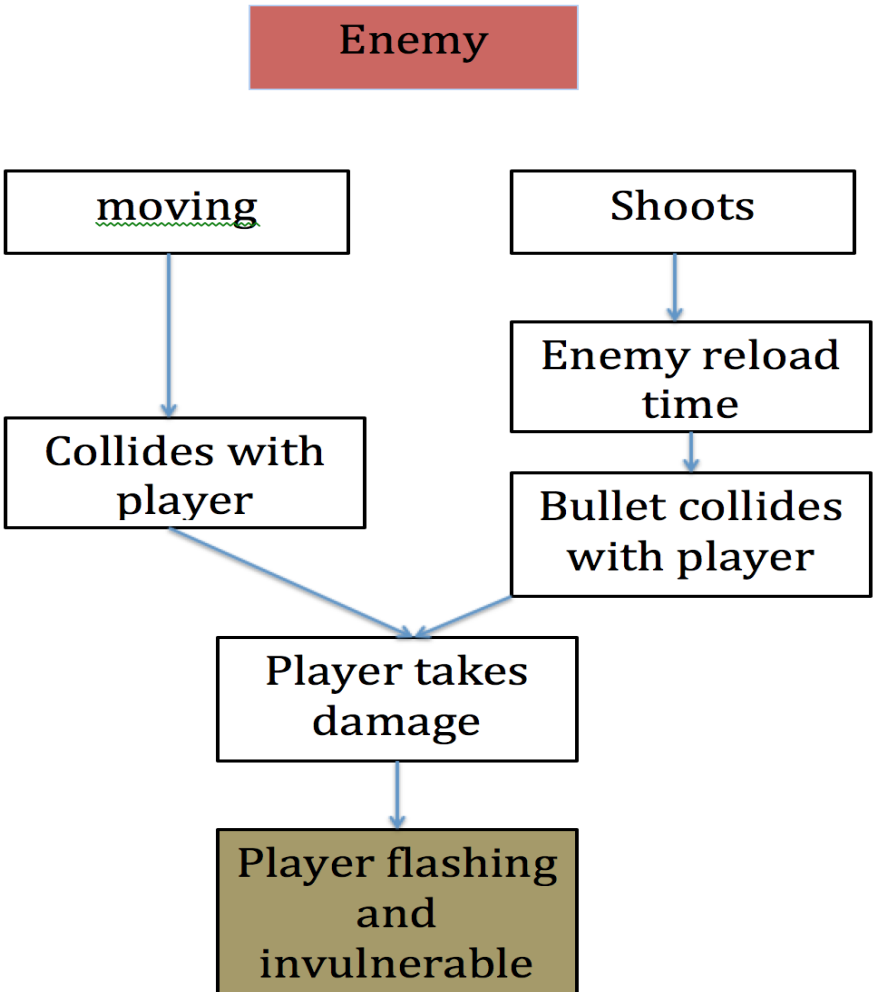
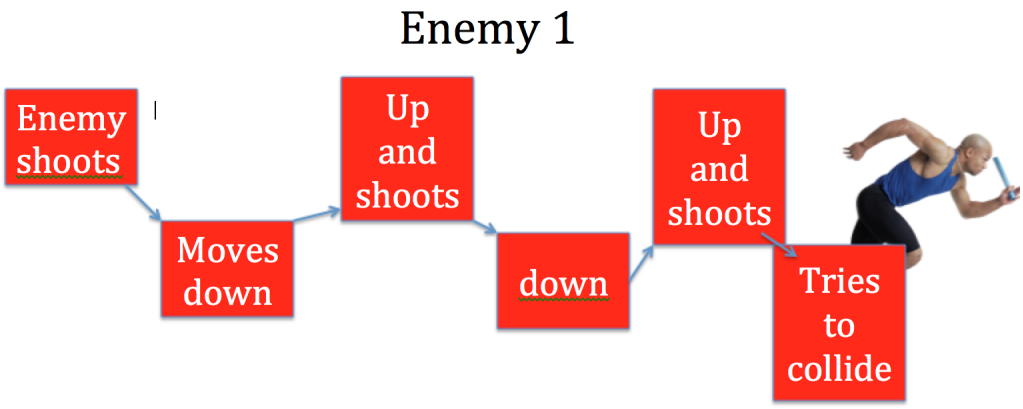


Figure 1

Figure 2



This type of enemy would move up and down to evade bullets from the player, it would then shoot and carry on moving trying to collide with the player.

Rarity: common
Health: 2 shots
Speed: fast
Abilities: none
Drops: ammo

Figure 3


init
Main loop Start
Model
View
Controller
Shutdown/ restart.
The Model, View Controller type architecture is an order of functions that are called in the Application's Main Loop, this is how Apps communicate with the User
or
Model
checks if exit button true.
This updates changes in objects or classes, this is where a move to the left may be cancelled by Collision or Ammo amount is called by hud so it will display - 1 Ammo
checking or Collisions against enemy bullets.
assuming Shooting anim is @ frames 6 to 12
View
draw backdrop()
draw platforms()
draw players()
draw enemies()
draw bullets()
draw effects()
draw buttons()
draws Hud Last.
Control
detect Keys()
detect Mouse()
detect Win()
detect Lose()
if(key == 'SPACE'){
P1.setJump(true);
}
or
if(mouseClick){
for(x){
if(hud.buttonArray.get(x).
CollidingWith(mouseX, mouseY)){
hud.buttonArray.get(x).
setBool(true);
}
}

Figure 4

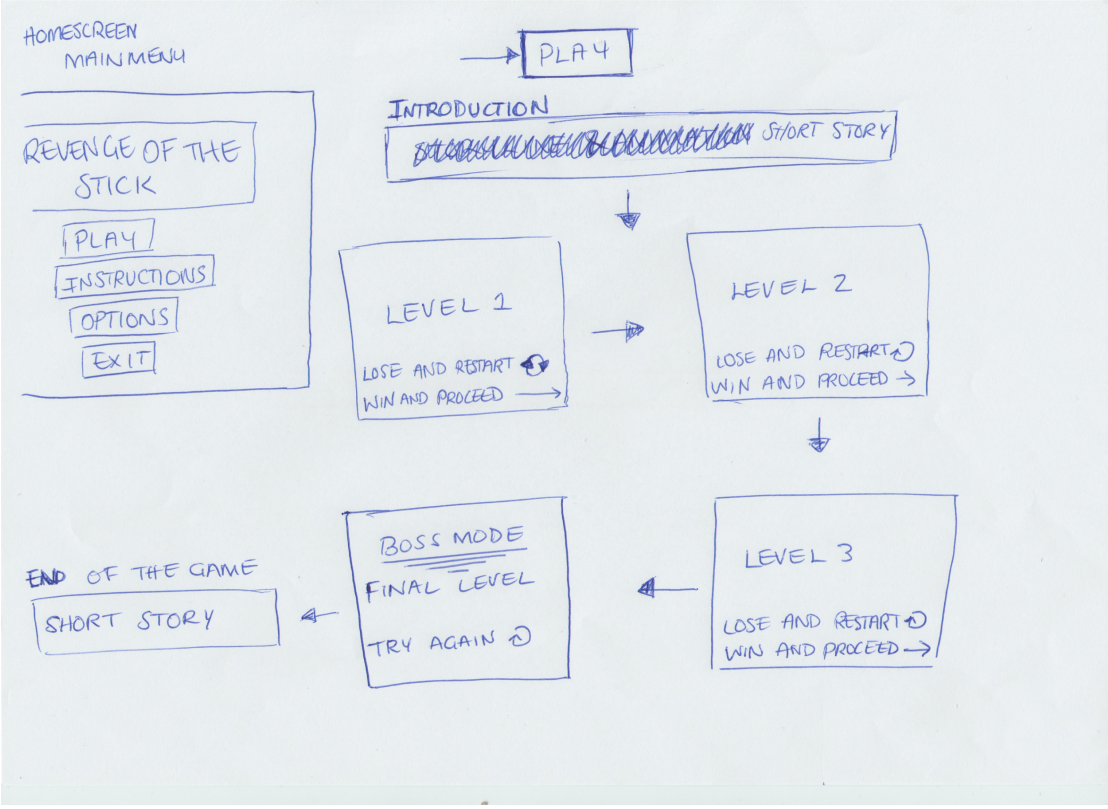

HOMESCREEN
MAIN MENU
REVENGE OF THE STICK
PLAY
INSTRUCTIONS
OPTIONS
EXIT
PLAY
INTRODUCTION
SHORT STORY
LEVEL 1
LOSE AND RESTART
WIN AND PROCEED
LEVEL 2
LOSE AND RESTART
WIN AND PROCEED
LEVEL 3
LOSE AND RESTART
WIN AND PROCEED
BOSS MODE
FINAL LEVEL
TRY AGAIN
END OF THE GAME
SHORT STORY

Figure 5

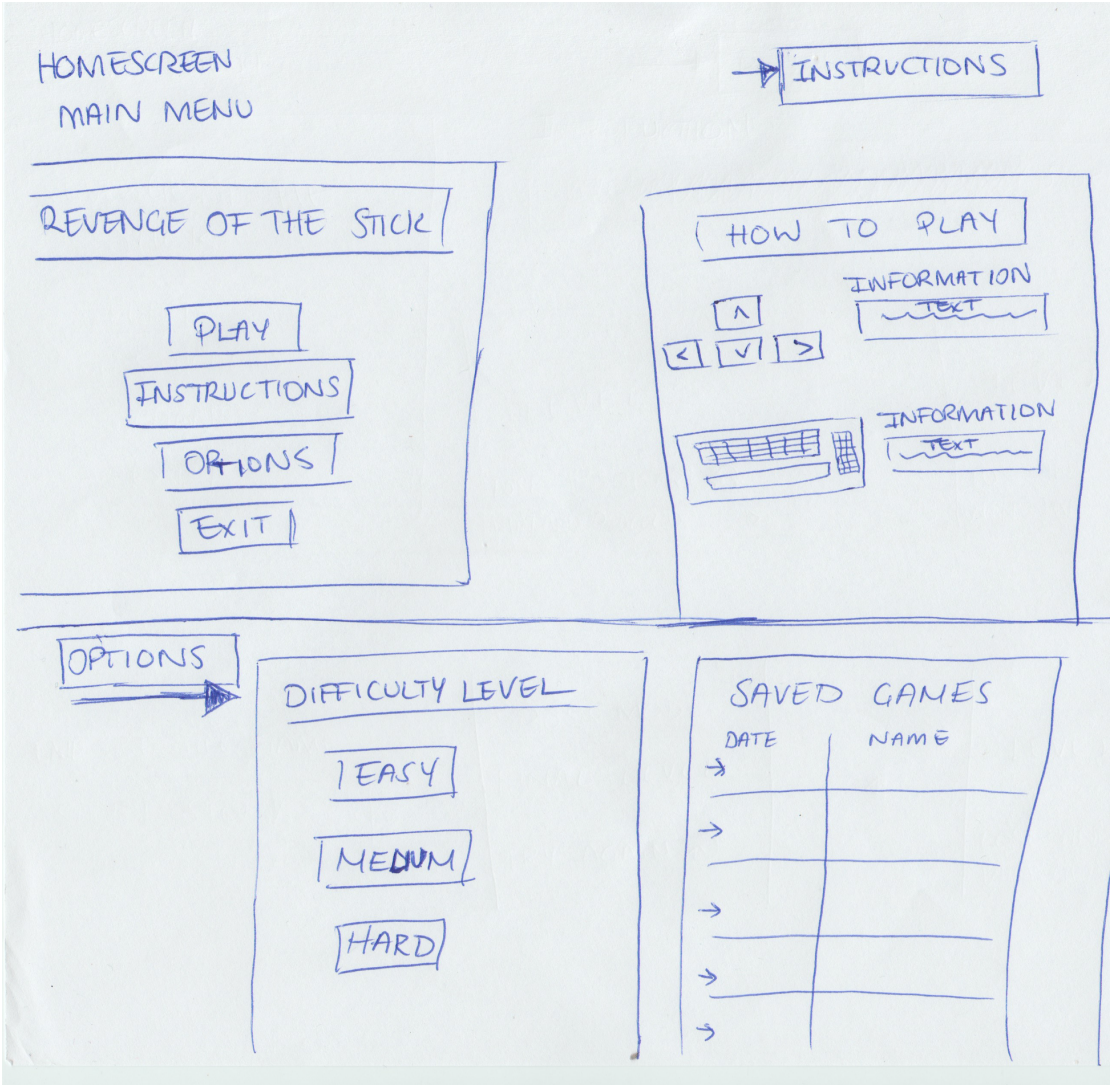


Figure 6

System requirement specification

Use case: getOptions
ID: 1
Brief description: The user goes to the options menu where there are resolution and buttons they can change to their preference.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is on the games main menu
Main flow: **This use case starts when the user accesses the options feature that allows them to alter the resolution and buttons in the game.** **1. The game has options in the main menu when it starts up, the user accesses "Options" from the menu.** **2. The user enters the options where they can change resolution and/or buttons to**

their own liking. **3. The game verifies user's choices and asks if it as they wanted.** **4. User clicks “Ok”** **5. The game changes resolution and /or buttons to their choices that were made.**
Post conditions: The user has configured the resolution/button
Alternative flow: **1. Incorrect resolution/button**
ID: 1.1
Brief description: The game informs the user that the resolution/button they have entered is not possible.
Primary actor: User
Secondary actor: None
Preconditions: The user has entered an invalid resolution/button to configure
Alternative flow: **The alternative flow starts after step 4 of the main flow.** **1. The system informs the user that the resolution/button they have entered are not viable as their computer screen does not have that resolution or the same button is being used twice to control the character.**
Post conditions: The user is then prompted again to re-choose their resolution/button preferences.
Alternative flow: **2. Back**
ID: 1.2
Brief description: The game has a back option to go back to the main menu, from the options menu.
Primary actor: User
Secondary actor: None
Preconditions: The user has entered into the games options menu.
Alternative flow: **The alternative flow starts after step 1 of the main flow.** **1. The game has the option to go back to the main menu, at any point, while in the options menu.**
Post conditions: The user is sent back to the main menu of the game.

Use case: getHighScore
ID: 2
Brief description: The user can access the high score section to see who has the best high score of the game.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is on the games main menu
Main flow: **This use case starts when the user accesses the high score feature that allows them to see who has a high score.** **1. The game has options in the main menu when it starts up, the user accesses "High Scores" from the menu.** **2. The user enters the high score where they can see who has the highest score.**
Post conditions: The user has checked high scores of the game
Alternative flow: **1. Back**
ID: 2.1
Brief description: The game has a back option to go back to the main menu, from the high score menu.
Primary actor: User
Secondary actor: None
Preconditions: The user has entered the high score menu
Alternative flow: **The alternative flow starts after step 1 of the main flow.** **1. The game has the option to go back to the main menu, at any point, while in the high score menu.**
Post conditions: The user is sent back to the main menu of the game.

Use case: Start
ID: 3
Brief description: The user can click start to play the game.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is on the games main menu
Main flow: **This use case starts when the user accesses the Start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses "Start" from the menu.** **2. The user enters the Start option and begins to play the game.**
Post conditions: The user has started playing the game.

Use case: moveLeft
ID: 4
Brief description: The user can make the character in the game move left when hitting the left key.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game.
Main flow: **This use case starts when the user accesses the start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses "Start" from the menu.** **2. The user enters the game and presses the left key.**
Post conditions: The user has made the character move left.

Use case: moveRight
ID: 5
Brief description: The user can make the character in the game move right when hitting the right key.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game.
Main flow: **This use case starts when the user accesses the start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses "Start" from the menu.** **2. The user enters the game and presses the right key.**
Post conditions: The user has made the character move right.

Use case: playerJump
ID: 6
Brief description: The user can make the character in the game jump when hitting the up key.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game.
Main flow: **This use case starts when the user accesses the start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses "Start" from the menu.** **2. The user enters the game and presses the up key.**
Post conditions: The user has made the character jump.
Alternative flow: **1. playerHang**
ID: 6.1
Brief description: The user can make the character in the game jump when hitting the up key and if it doesn't make the entire jump properly on to a platform, the character hangs on platform.
Primary actors: User
Secondary actors: None

Preconditions: The user has jumped and landed next to the edge of a platform.
Alternative flow: **This use case starts when the user accesses the start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses "Start" from the menu.** **2. The user enters the game and presses the up key to land on a platform.** **3. The character is hanging on the edge of the platform.** **4. The user presses the up directional button for a second time to get on the platform properly**
Post conditions: The user has made the character get on the platform.

Use case: playerShoot
ID: 7
Brief description: The user can make the character in the game shoot when hitting the space bar.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game.
Main flow: **This use case starts when the user accesses the start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses "Start" from the menu.** **2. The user enters the game and presses the space bar.**
Post conditions: The user has made the character shoot.

Use case: Bullets
ID: 8
Brief description: The user can make the character pick up bullets found in the game.
Primary actors: User
Secondary actors: None

Preconditions: The user has opened up the game and is playing the game.
Main flow: **This use case starts when the user accesses the start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses "Start" from the menu.** **2. The user enters the game and walks over bullets.**
Post conditions: The user has made the character increase their ammunition.
Alternative flow: **1. No ammunition gained**
ID: 8.1
Brief description: The character in the game did not receive an increase in ammunition.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game and has walked over bullets.
Alternative flow: **The alternative flow starts after step 2 of the main flow.** **1. The game did not recognise the bullets and did not pick it up.** **2. Walk back and forth on the bullets to pick it up.**
Post conditions: Bullets recognised and picked up, characters ammunition increased.

Use case: Guns
ID: 9
Brief description: The user can make the character pick up various guns found in the game.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game and has found a gun lying on the ground.
Main flow: **This use case starts when the user accesses the start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses**

"Start" from the menu. **2. The user enters the game and finds a gun and walks over it.**
Post conditions: The user has made the character get a new gun.
Alternative flow: **1. No guns gained**
ID: 9.1
Brief description: The character in the game did not receive a new gun.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game and has walked over gun.
Alternative flow: **The alternative flow starts after step 2 of the main flow.** **1. The game did not recognise the gun and did not pick it up.** **2. Walk back and forth on the gun to pick it up.**
Postconditions: Gun recognised and picked up, character has earned a new gun.

Use case: healthIndicator
ID: 10
Brief description: The user can see the character's health in the indicator on the top left hand side of the screen.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game.
Main flow: **This use case starts when the user accesses the start feature that allows them to play the game.** **1. The game has options in the main menu when it starts up, the user accesses "Start" from the menu.** **2. The user enters the game.** **3. When character is hurt by enemies, the health decreases and is shown by the health indicator.**
Post conditions: The user has made the characters health decrease.

Use case: calculateScore
ID: 11
Brief description: The user can make the character shoot enemies which are each worth a certain amount of points.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and is playing the game and is shooting enemies.
Main flow: This use case starts when the user accesses the start feature that allows them to play the game. 1. The game has options in the main menu when it starts up, the user accesses “Start” from the menu. 2. The user enters the game and shoots enemies. 3. Each particular enemy is worth a certain amount of points.
Post conditions: The user has made the character gain points to be added to their score.

Use case: gameOver
ID: 12
Brief description: The user lets the character die as it lost too much health and the game is over.
Primary actors: User
Secondary actors: None
Preconditions: The user has opened up the game and has been killed by enemies.
Main flow: This use case starts when the user accesses the start feature that allows them to play the game. 1. The game has options in the main menu when it starts up, the user accesses “Start” from the menu. 2. The user enters the game and shoots enemies. 3. Each particular enemy is worth a certain amount of points. 4. The character dies and the score is calculated. 5. The game prompts the user to insert 3 letters to signify their score. 5. The score is then added to the high score database
Post conditions: The user has their score added to the high score database.

Figure 7

Non-functional Requirements

Performance
The waiting time while the game starts up, as well as accessing information such as high scores should be minimal. It shouldn't take up much of the CPU to play either.
Reliability
The game shall be available 24 hours a day, 7 days a week without any issues. The game shall always provide real time information about high scores.
Availability
The game shall be available to anyone and everyone who would like to play it. They can play online or download it.
Usability
The game shall provide a GUI which is easily navigable. It shall be very easy to navigate through the menus and sub-menus to give users control and access. The game shall have simple moves accessed by certain buttons which the users will have full control of.
Integrity
The game should be secure and must use encryption to protect the game as well as the high score database. System administrators will only have access to change or remove users. Users need to be authenticated before having access to any personal data.
Portability
The game must be portable; it should be able to run on any operating system such as Windows 32bit or OSX. The software must able to be installed on portable devices and should be able to run on different computers.
The Look and Feel
The game must have a nostalgic feel to it, it should feel similar to 1980's Mario, platform style game. The game must not be a copy of any other game.
Maintainability
The game and database should be maintained regularly. Maintaining of these should be done by the system administrator.
Recoverability
If for some reason, something goes wrong with the game or any of its high score data, the data should be recovered easily.

Requirement Name: Performance	**Requirement Type: NFR**	**Event/Use Case**
Description: The waiting time while the game starts up, as well as accessing information such as high scores should be minimal. It shouldn't take up much of the CPU to play either.		
Rationale: For the users to be able to play without waiting long, it should run smoothly without any lags or glitches.		
Source:		

Fit Criteria: The game should run without any glitches and input response (when a button is hit for an action in the game) should be instantaneous. The game should be available for service when requested by the end-users. The throughput requirements (how much the system can accomplish within a specified amount of time) should be faster, so that there is no time wasted for the users.				
Customer Satisfaction:		**Customer Dissatisfaction:**		
Priority: Essential		**Conflicts:**		
Supporting Material:			**Volere** **Source: Atlantic Systems Guild**	
History: Created December 12, 2015				

Requirement Name: Reliability	**Requirement Type: NFR**	**Event/Use Case**
Description: The game shall be available 24 hours a day, 7 days a week without any issues. The game shall always provide real time information about high scores.		
Rationale: For the users to be able to access the game whenever they want, and are able to check high scores. The ability of the game to deliver services as specified.		
Source:		
Fit Criteria: The game must always be online and the users must always be able to connect to the system and in any case of disorder or maintenance issue, the admin should take no more than 30 min to get it back online.		
Customer Satisfaction:	**Customer Dissatisfaction:**	
Priority: Essential	**Conflicts:**	
Supporting Material:	**Volere** **Source: Atlantic Systems Guild**	
History: Created December 12, 2015		

Requirement Name: Availability	**Requirement Type: NFR**	**Event/Use Case**
Description: The game shall be available to anyone and everyone who would like to play it. They can play online or download it.		
Rationale: The ability of the system to deliver services when requested.		
Source:		
Fit Criteria: The system should be available when requested 24 hours a day, 7 days a week, to users and administrators. The high score database must only be available to the system administrator.		
Customer Satisfaction:	**Customer Dissatisfaction:**	
Priority: Essential	**Conflicts:**	
Supporting Material:	**Volere** **Source: Atlantic Systems Guild**	
History: Created December 12, 2015		

Requirement Name: Usability	**Requirement Type: NFR**	**Event/Use Case**
Description: The game shall provide a GUI which is easily navigable. It shall be very easy to navigate through the menus and sub-menus to give users control and access. The game shall have simple moves accessed by certain buttons which the users will have full control of.		
Rationale: The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of game. The GUI should be easily navigable.		
Source:		
Fit Criteria: To provide the users a simple and easy way to access the game and stop them		

from getting confused about how to play it.				
Customer Satisfaction:		**Customer Dissatisfaction:**		
Priority: Essential		**Conflicts:**		
Supporting Material:			**Volere** **Source: Atlantic Systems Guild**	
History: Created December 12, 2015				

Requirement Name: Integrity	**Requirement Type: NFR**		**Event/Use Case**	
Description: The game should be secure and must use encryption to protect the game as well as the high score database. System administrators will only have access to change or remove users. Users need to be authenticated before having access to any personal data.				
Rationale: The game shall protect its data from incorrect and prevent users from changing the data. It shall also prevent users from causing intentional abuse.				
Source:				
Fit Criteria: So that the data of the high scores are protected and is not released to the users to edit. To ensure the integrity of the system from accidental or malicious damage				
Customer Satisfaction:		**Customer Dissatisfaction:**		
Priority: Essential		**Conflicts:**		
Supporting Material:			**Volere** **Source: Atlantic Systems Guild**	
History: Created December 12, 2015				

Requirement Name: Portability	Requirement Type: NFR	Event/Use Case
Description: The game must be portable; it should be able to run on any operating system such as Windows 32bit or OSX. The software must able to be installed on portable devices and should be able to run on different computers.		
Rationale: The ease with which the game can be transferred from one environment to another. The game must be able to be installed on various operating systems and should be able to run on different computers and devices.		
Source:		
Fit Criteria: So that the users can carry the game with them on portable device like a mobile phone or tablet and be able to use the game from anywhere where internet access is available.		
Customer Satisfaction:	**Customer Dissatisfaction:**	
Priority: Low	**Conflicts:**	
Supporting Material:	**Volere** **Source: Atlantic Systems Guild**	
History: Created December 12, 2015		

Requirement Name: Maintainability & Recoverability	Requirement Type: NFR	Event/Use Case
Description: The game and database should be maintained regularly. If something goes wrong with the game or any of its high score data, the data should be recovered easily.		
Rationale: The ability to change the game to deal with new technology or to fix defects. The ability to recover any lost data without a major problem.		
Source:		
Fit Criteria: Maintaining of the game and database should be done by the system admin only. If for some reason, something goes wrong with the system, the data should be recovered easily.		

Customer Satisfaction:		**Customer Dissatisfaction:**		
Priority: Essential		**Conflicts:**		
Supporting Material:			**Volere** **Source: Atlantic Systems Guild**	
History: Created December 12, 2015				

Figure 9

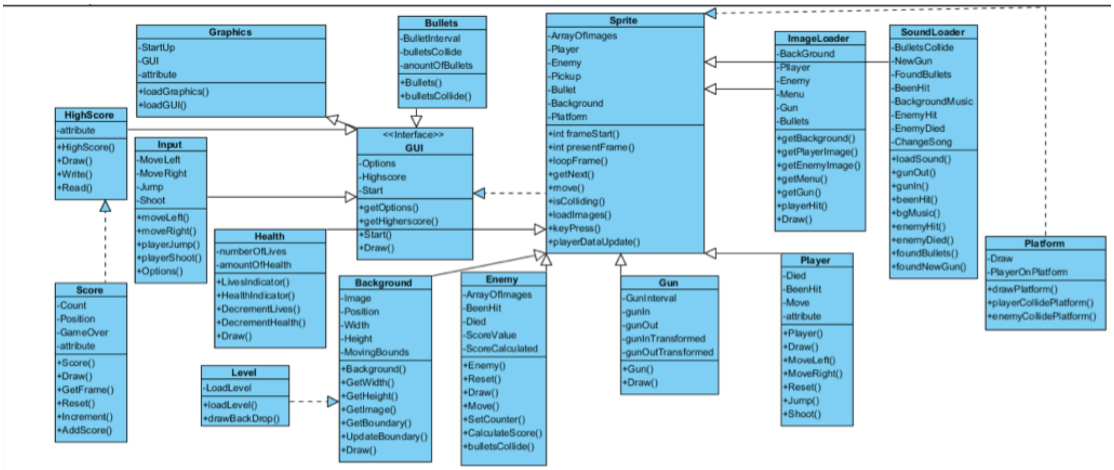
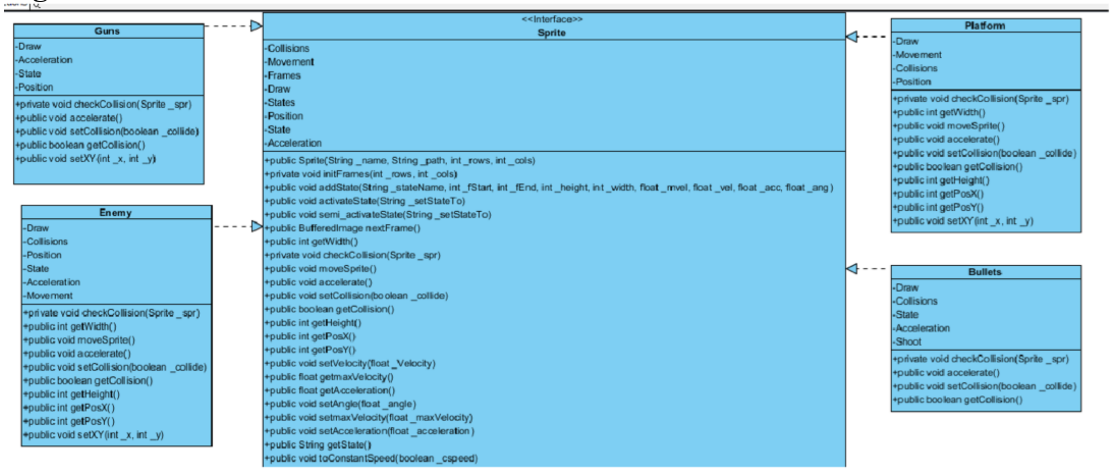


Figure 10



Distribution of Marks:

Our 5th member Nikhil was not here for the entire 2nd or 3rd term, he contributed nothing, no communication, did not come into uni for meetings or lectures, therefore we have all agreed as a group to give him 0%.

Jethro : 100%
Arman : 100%
Nayim Uddin (Mohammed): 100%
Abraham: 100%
Nikhil: 0%