|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Course: ICS3U1-03  Last Updated: June 15, 2016 |  | Date Started: June 1st, 2016  Date Delivered: June 16, 2016 |  | Authors: Adrian Pasqualini  Timothy Quijano |



Aquatic Attack

Computer Science CPT

Table of Contents

[Customer Requirements 2](#_Toc453913391)

[Project Timeline 3](#_Toc453913392)

[Design Proposal 4](#_Toc453913393)

[Design Proposal 5](#_Toc453913394)

[Implementation Details and Deliverables 6](#_Toc453913395)

[Implementation Details and Deliverables 7](#_Toc453913396)

[Maintenance 9](#_Toc453913397)

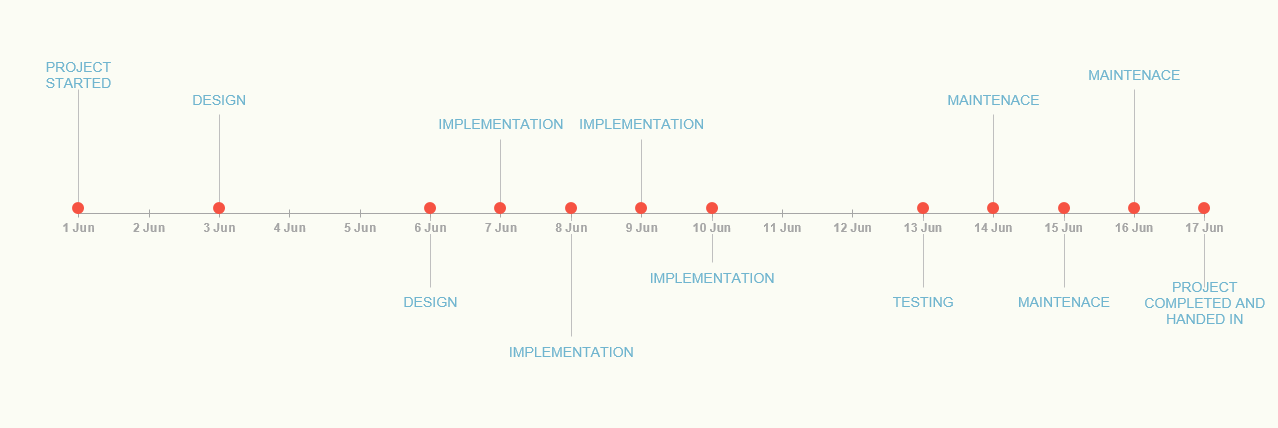
[Maintenance 10](#_Toc453913398)

# Customer Requirements

|  |  |
| --- | --- |
|  | The purpose of this culminating task is to demonstrate programming abilities through the production of an arcade game using the module pygame. Within the program, proper use of comments, loops, classes, subprograms, methods, and drawing images, as well as other tools, are necessary. Students must be able to use their knowledge of functions that were used throughout the course, with an addition of individual research. In addition to the game, formalized testing and a project report must also be submitted, in order for the teacher to understand the concept of the student’s project.  Groups of two are allowed to encourage collaboration, thus producing a greater product. However, both students must actively participate and work together to complete the task. |



# Project Timeline



Timeline of events that occurred on each day

**Actual Timeline:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | 01-Jun | 03-Jun | 06-Jun | 07-Jun | 08-Jun | 09-Jun | 10-Jun | 13-Jun | 14-Jun | 15-Jun | 16-Jun |
| **Plan** | *Reqs* | *Design* | *Design* | *Imp* | *Imp* | *Imp* | *Imp* | *Test* | *Main* | *Main* | *Main* |

**Planned timeline:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date** | 01-Jun | 03-Jun | 06-Jun | 07-Jun | 08-Jun | 09-Jun | 10-Jun | 13-Jun | 14-Jun | 15-Jun | 16-Jun |
| **Plan** | *Reqs* | *Design* | *Design* | *Design* | *Imp* | *Imp* | *Imp* | *Imp* | *Test* | *Test* | *Main* |

|  |  |
| --- | --- |
|  | The predetermined schedule differed slightly because designing the game took less time than we thought. Adrian and Timothy collaborated at home, thinking of ideas and how to apply them. As a result, we had more time for implementation. This additional day was beneficial because creating the code was a little harder than we imagined.  After we made the game, it was played by other students during the day in order to catch any “bugs”. Because we did so, we were able to identify and fix problems, as well as use the feedback to further enhance the gaming experience. We had a lot of time to fix problems because we were able to identify problems through people playing the game. |

# Design Proposal

**UMLs**

|  |  |
| --- | --- |
| **Wall Class** | |
| **Attributes** | |
| Int. | X – coordinate |
| Int. | Y – Coordinate |
| Int. | Width |
| Int. | Height |

|  |  |  |
| --- | --- | --- |
| **Fish Class** | | |
| **Attributes** | | |
| Surface | | Character Image |
| (Int. Int.) | | Rectangle around image |
| Int. | | X – Coordinate |
| Int. | | Y – Coordinate |
| Int. | | Max Speed |
| Int. | | Min. Speed |
| Int. | | Health |
| Int. | | Player Number |
| Int. | | Damage |
| **Behaviors** | | |
| Void | Update | |

|  |  |
| --- | --- |
| **Spike Class** | |
| **Attributes** | |
| Surface | Spike Image |
| (Int. Int.) | Rectangle around image |
| X – Coordinate | 625 |
| Y - Coordinate | 23 |
| Int. | Direction |
| **Behaviors** | |
| Void | Update |

|  |  |  |
| --- | --- | --- |
| **Bubble Class** | | |
| **Attributes** | | |
| Surface | | Spike Image |
| (Int. Int.) | | Rectangle around image |
| Int. | | X – Coordinate |
| Int. | | Y – Coordinate |
| String | | Direction |
| **Behaviors** | | |
| Void | Update | |

|  |  |  |
| --- | --- | --- |
| **Power-up Class** | | |
| **Attributes** | | |
| Surface | | Character Image |
| (Int. Int.) | | Rectangle around image |
| Int. | | X – Coordinate |
| Int. | | Y – Coordinate |
| **Behaviors** | | |
| Void | Update | |

|  |  |  |
| --- | --- | --- |
| **Pillar Class** | | |
| **Attributes** | | |
| Surface | | Character Image |
| (Int. Int.) | | Rectangle around image |
| Int. | | X – Coordinate |
| Int. | | Y – Coordinate |
| **Behaviors** | | |
| Void | Update | |

# Design Proposal

**State Diagrams**

|  |  |
| --- | --- |
| **Wall:** Wall\_Up | |
| **Attribute** | **Value** |
| X – coordinate | 0 |
| Y – Coordinate | -5 |
| Width | 1440 |
| Height | 5 |

|  |  |
| --- | --- |
| **Fish:** Puffer | |
| **Attribute** | **Value** |
| Character Image | ‘puffer.png’ |
| Rectangle around image | (74, 62) |
| X – Coordinate | 800 |
| Y – Coordinate | 400 |
| Max Speed | 9 |
| Min. Speed | -9 |
| Health | 250 |
| Player Number | 1 |
| Damage | -25 |

|  |  |
| --- | --- |
| **Spike:** Spike12 | |
| **Attribute** | **Value** |
| Spike Image | ‘spike12.png’ |
| Rectangle around image | (23, 39) |
| X – Coordinate | 625 |
| Y - Coordinate | 23 |
| Direction | ‘12’ |

|  |  |
| --- | --- |
| **Bubble:** Bubble(right) | |
| **Attribute** | **Value** |
| Spike Image | ‘bubble.png’ |
| Rectangle around image | (22, 22) |
| X – Coordinate | 455 |
| Y - Coordinate | 420 |
| Direction | “right” |

|  |  |
| --- | --- |
| **Power-up:** Damage Boost | |
| **Attribute** | **Value** |
| Character Image | ‘red\_left’ |
| Rectangle around image | (88, 85) |
| X – Coordinate | 2440 |
| Y – Coordinate | 650 |

|  |  |
| --- | --- |
| **Pillar:** Pillar | |
| **Attribute** | **Value** |
| Character Image | ‘pillar’ |
| Rectangle around image | (65, 300) |
| X – Coordinate | 320 |
| Y – Coordinate | 522 |

# Implementation Details and Deliverables

|  |  |
| --- | --- |
|  | Aquatic Attack was created using the pygame module. All of the techniques used were acquired in the course, with the small addition of a few others. The two-player game is a battle of aquatic animals, a puffer fish and a seahorse. Each player plays either character and uses their respective abilities to eliminate the other. Once they accomplish that task, they have the option to continue playing or end the game.  The characters are instances of a Fish class that inherited from the sprite class. Creating this as a child class allows us to process collisions. This was helpful with creating the boundaries of the screen, so the characters cannot pass the top or bottom, and travel from side to side. We checked if the characters collided with instances of the Wall class, maintaining or changing their X or Y coordinate, thus controlling their movement on the screen. Moreover, collisions were used to process the projectiles hitting the players. Similar to the walls, we checked if a fired projectile made contact with a character. If this occurred, damage was dealt to the other player’s health points.  To easily process collisions, we put all of the projectiles and walls into lists and checked if certain objects made contact with objects in the list. For instance, for every spike from the spike list that made contact with the pufferfish, deal damage to the pufferfish. Similarly, for every bubble shot from the seahorse, deal damage to the puffer fish.  The seahorse does not move as fast as the pufferfish. However, it can make turns easier. To understand this, one must know how the movement of the characters work. Each character’s X and Y coordinate was affect by another one of its attributes, speed. If the left arrow key was pushed, the X axis speed of the pufferfish decreased. This meant that the pufferfish moved left. However, there is a maximum value at which the decreasing stops. Furthermore, the speed is divided when the key is released.  The seahorse’s maximum value for moving is less than that of the pufferfish’s, meaning that the seahorse is slower. However, the seahorse’s dividing value is greater than that of the pufferfish’s, meaning that it can make turns more quickly.  The user may press spacebar for the seahorse to create a bubble and add it to the bubble list, however a bubble instance will only be created if there are no other bubbles existing. When creating a bubble, the sea horse’s direction was passed into the instance so that the projectile flies in the right direction. When the bubble collides with the pufferfish, or wall, it is removed from the bubble list and another one may be created.  Now onto the pufferfish. This character deals damage when simply touching the pufferfish. At first glance this may seem overpowering, however it is not because the seahorse can dodge because of its greater agility. The pufferfish may also fire eight spikes going in all directions, dealing damage to the seahorse upon contact. Although, this ability has a drawback. It no longer harms the seahorse for a certain time; the pufferfish deflates. Similar to the bubbles from the seahorse, the spikes are created when no other spikes in the spikes list exist. Once each spike makes contact with an object, it disappears from both the screen and the spikes list.  There are other fish in the game- “powerups”. Collecting these instances of the Powerup class, that inherited from the sprite class so that collision is possible, allows for the player to deal double the damage for a limited amount of time. This timer is displayed at the top of the screen, along with the score count. This count is increased for whomever wins the round. |

# *C:\Users\Tim\AppData\Local\Microsoft\Windows\INetCache\Content.Word\seahorse_right.png*Maintenance



New Seahorse

|  |  |
| --- | --- |
|  | Aquatic Attack has been played many times by friends and family towards the end of the project once it was almost done. We were able to collect valuable feedback that improved the gaming experience, as well as find bugs that inhibited the gameplay. Because of our game-testers, we were able to identify problems that were not apparent to us initially. The original seahorse got negative comments- that it was unfitting to the rest of the artwork. To accommodate, we replaced the image with a more cartoon-like figure that matched the background. |

Old Seahorse

|  |  |
| --- | --- |
|  | Overall, the production of the game was both an enjoyable learning experience. Adrian and I are good friends who’ve already worked on previous projects. We already know each other’s working habits and are accustom to them. To us, this experience is yet another positive because of our collaborative efforts.  We are most proud of the movement of the characters. Adrian spent a large period of time learning how the values change with specific techniques. As a result of great work, our users are able to manipulate their players with comfort and ease. Turning does not seem unreal and the players gradually slow down, mimicking a real-world scenario. With the addition of the differentiating speeds and maneuverability, the movement of characters is what we are proudest of.  Aside from the controlling of characters, we are also content with the pufferfish’s ability to explode. It took us some work to get it operational, however it was worth it. As aforementioned, the puffer fish deals damage when in contact with the seahorse. However, it can shoot spikes, causing it to contract and no longer deal damage upon contact. It was a slightly difficult to change the appearance for a timed amount, while also deleting the spike upon contact. We take pride in our puffer fish’s ability.  An area of improvement would be the graphics of our game. We could enhance the gaming experience by creating a more detailed background and having objects in the scene collide with players. For instance, the seahorse cannot pass through a sunken ship. Moreover, we could also improve the appearance of the health bars and fonts, making them more pleasing to the eye. |

# Maintenance

|  |  |
| --- | --- |
|  | Aquatic Attack has been played many times by friends and family towards the end of the project once it was almost done. We were able to collect valuable feedback that improved the gaming experience, as well as find bugs that inhibited the gameplay. Because of our game-testers, we were able to identify problems that were not apparent to us initially. The original seahorse got negative comments- that it was unfitting to the rest of the artwork. To accommodate, we replaced the image with a more cartoon-like figure that matched the background. |

|  |  |  |
| --- | --- | --- |
| **Task** | **Responsibility** | **Description** |
| Mr. Reid | ProjectTemplate | Constructed a project report template for the students to use to complete final report. |
| Adrian | Movement | Created the movement of the characters. |
| Adrian | Created the walls | Created the boundaries. |
| Adrian | Processed the events | Handled what would happened if keys were pressed. |
| Adrian | Powerup | Created and coded the powerup features. |
| Adrian | Commenting | Documented the code, with Timothy inspecting it afterwards. |
| Timothy | Pictures | Found images that were used and formatted them for usage (made backgrounds invisible and resize). |
| Timothy | Report | Wrote and formatted the report and flowcharts. |
| Timothy and Adrian | Puffer fish exploding | Timothy began to create the logic for this special ability, however Adrian greatly enhanced it with smarter logic. |
| Timothy and Adrian | Background | Timothy created the background, with some additions made by Adrian. |