Project X – Two player Competitive Shooter Game

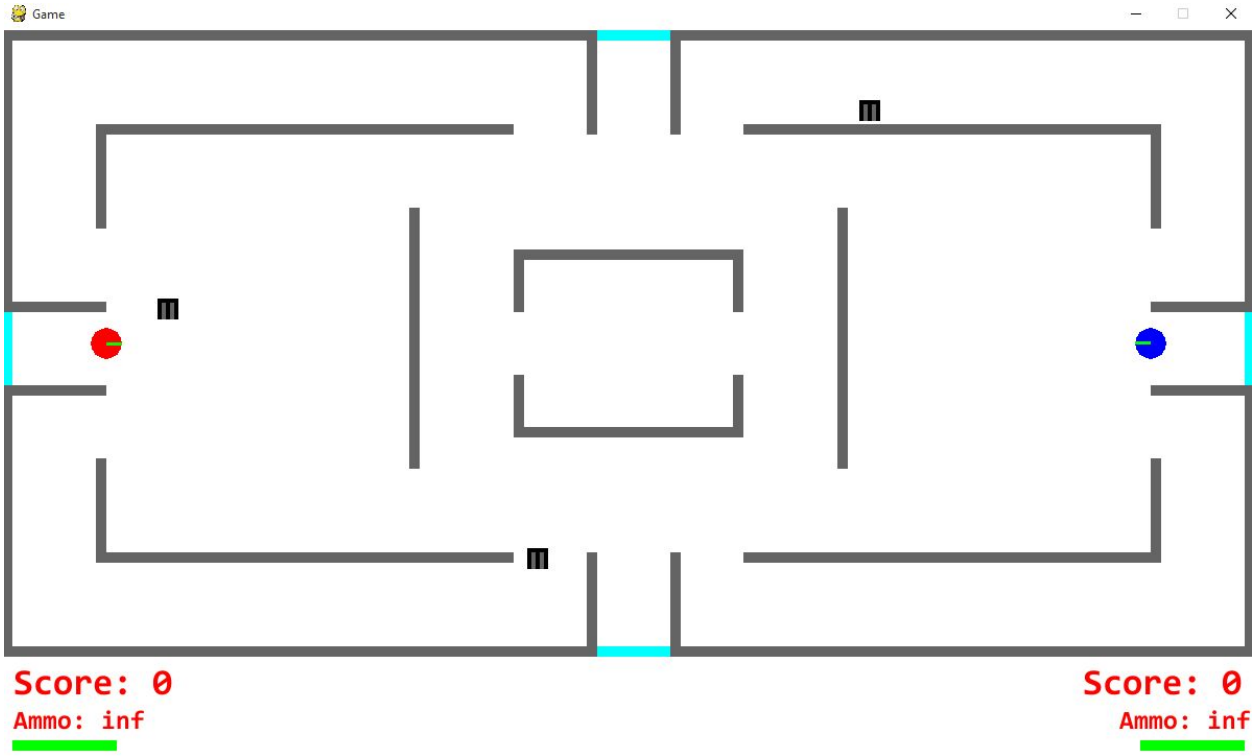


Figure 1.0

Authors: Aniekan Umoren & Conner Tenn
Date Started: Thursday, December 17, 2015
Course: ICS-3U
Last Updated: Wednesday, January 20, 2016
Date Delivered: Friday, January 22, 2016

Customer Requirements:
Monday, January 4, 2016
12:00 - 13:30

The task given by our client during the conference was to create a top down, multiplayer, Shoot 'Em Up game. This game is to be suitable for all customers 10 years and up, containing quick paced addictive gameplay. Elements to be included are a diverse collection of power-ups and abilities, 2-player capabilities, and a soundtrack that match the nature of the game.

Project Timeline:

	Requirements	**Design**	**Implementation**	**Testing**	**Deployment**
Proposed	Jan 4-6	Jan 5-7	Jan 8-12	Jan 13-15	Jan 18-20
Completed	Jan 4-5	Jan 5-8	Jan 6-14	Jan 15-17 (over the weekend)	Jan 18-20

Figure 1.1: This represents the data of our proposed schedule versus the actual time spent on certain stages.

The reason behind the deviations from the proposed schedule is that, coming up with solid mechanics for the game and designing their function, took a bit longer than anticipated. However, the implementation step was relatively straight forward as a result. Little time was spent on the testing stage because not many subprograms returned any tangible results for testing. This is to be expected as much of the programs processes return more complex forms of output which can not be easily simulated in a script. Note that some of the stages overlap.

Design Proposal:

The product is to be created using Pygame module, taking full advantage of its computer graphics and sound libraries. However, most of the project will be the original work of the authors. A notable feature of the pygame library that will be used is the collisions handling as it makes mechanics such as shooting your opponent,or teleportation possible.

The goal is to use the classes presented in Figure 1.2 in order to neatly manage the flow and structure of the program. No only does it increase the visual aesthetics of the code, but it also makes it easier to understand as it models the real world. For example, in Figure 1.2, behaviours like "move" and "shoot" are part of the PLayer class and were modelled after

their real-world counterparts. The shoot method creates a new bullet and the move method translates the players across the screen.

The full design of Project X (flowcharts, UML, state diagrams etc.) can be found in a separate document titles ‘Project X Design Doc’.

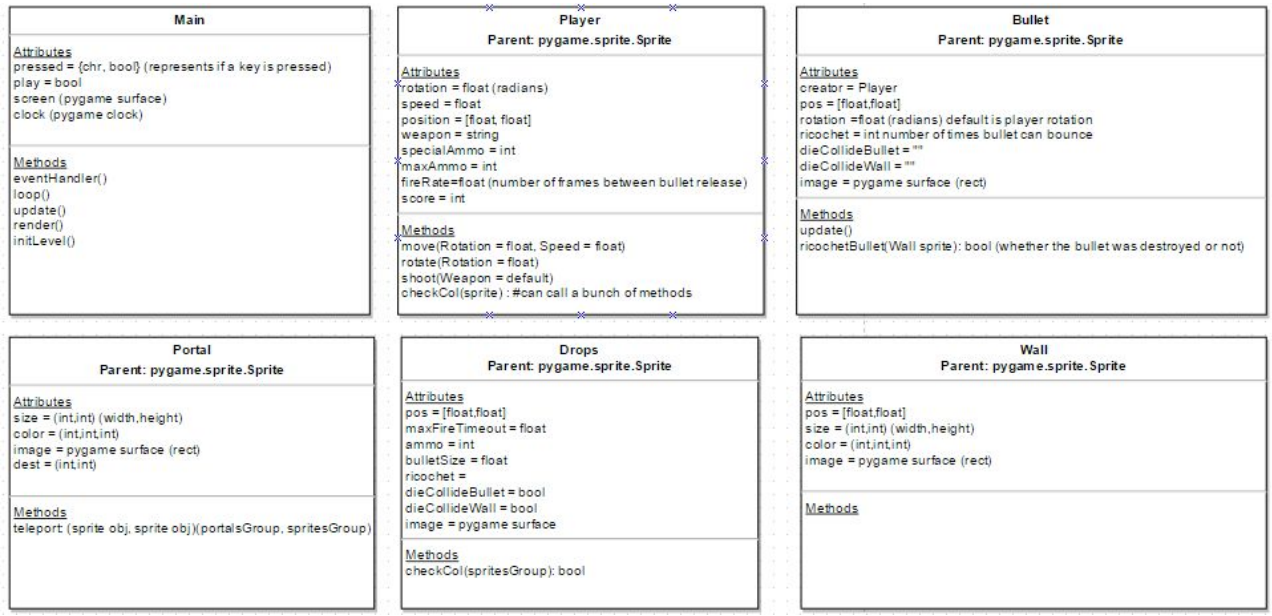
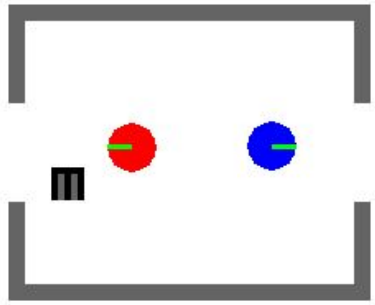


Figure 1.2: There are six classes, each possessing certain “attributes” and “behaviours” that govern the way the game operates.

Implementation and Deliverables:

The task of successfully creating a fun-filled game was accomplished; the success of this project can be attributed to the the programming knowledge provided by this course. For example, object orientation is a very key aspect our program, as every component of it, whether it be the player, the main program, or the portals are separated into neat "collections" of attributes and methods. This makes the handling of such large projects much easier.

In addition to object orientation, the use of selection ("conditional" code) and repetition can be found throughout the source code. Mechanics such as creating the level would be utterly tedious without such programming techniques. Other elements such as updating the players' positions or randomly spawning power-ups around the maps would be impossible.



A more detailed description of the game is as follows:
Project X is a two player game where player face off against each other for bragging rights and supremacy. The controls consist of the standard up, down, left and right, however, the players can also rotate independent of their direction of movement. This allows for more skilled and immersive gameplay.

Figure 1.3

This game is not an ordinary "Shoot 'Em Up" experience as there are special mechanics to keep the game interesting and and the players on their toes.

Item Boxes have the power to alter the abilities of a player's bullet giving them special capabilities such as blasting through walls, bouncing of walls, rapid fire, or some combination of all three.

Figure 1.4

Portals have the ability to transport both players and bullets to another randomly selected portal. This allows for quick transport around the map or can be used strategically to surprise your opponent.

Figure 1.5: A bullet about to be transported by the portal

Required Software:

To run this product visit the python website at *www.python.org/*, and install python 2.x. Next visit the Pygame website and to download the pygame module at *http://www.pygame.org/download.shtml*. Once the steps above are completed, you will now be able to enjoy a whole new gaming experience.

Maintenance:

The steps taken between the cultivation of an ideas and the full realization of Project X were not problem free (as they never), and were sometimes steps in the wrong direction. However, the moments of teamwork and diligence are what pulled the project through.

The first challenge presented itself during the design phase of the project. If you look through the design document, you will will find an immense flow chart representing one of our many methods and subprograms. Determining exactly how these methods would function was quite difficult in the beginning. As a result of this, the design stage wasn't done completely before implementation, thus limiting the amount of time that could be spent on it.

Another problem occurred during the implementation and was more of a collaboration issue. Often two different programmers will have two different "styles" of coding and they way the solve problems. This required getting used to each other's style of programming, and collaborating in a way that makes these differences useful. Although there were some problems with this, it improved with time. If we were given the opportunity to redo a certain aspect of the development process, it would have to be the maintenance of the report. Although we made efforts to keep it updated as we progressed, it ended being forgotten until the very end.

Despite all the challenges faced, there were moments of success that any developer would be proud of. A problem that task allocation creates, pertaining to the implementation of code, is the inevitable merging process that must take place. This process is slow, messy, and can end up breaking the program. Thankfully our disvovery of online IDE's such as *www.c9.io* and *www.floobits.com* got rid of the "middle man" by allowing us developers to work on the same project from different computers simultaneously. Another notable achievement was our ability to sufficiently comment as we went, thus reducing the workload at the end, and make the program easier to comprehend.

When looking back at the creation of any project, there will always be problems that presented themselves along the way but it is these challenges that help us to graduate from being mere “code monkeys”, to respected senior programmers. Besides, the end result is what makes it all worth it in the end.

Resources and Allocations:

Author	**Responsibility**	**Description**
Aniekan	Co-Level Design/Building	Formulating ideas for the level and creating it.
Conner	Co-Level Design	Formulating ideas for the physical structure of the level.
Aniekan	Report Management	Ensuring the report was in order.
Conner	Formalized Testing	Writing a script to provide a number of test cases, ensuring the proper functioning of the program.
Aniekan & Conner	Design	Constructing and visually representing the mechanics of the game through the use of flowcharts and UMLs.
Conner	Design Doc Formatting	Putting all the flowcharts, UML, and state diagrams into one tidy document.

Figure 1.6: A chart indicating how the tasks were allocated between the developers.

References:

https://pygame.org/wiki/RotateCenter?parent

```
def rot_center(image, angle):
    """rotate an image while keeping its center and size"""
    orig_rect = image.get_rect()
    rot_image = pygame.transform.rotate(image, angle)
    rot_rect = orig_rect.copy()
    rot_rect.center = rot_image.get_rect().center
    rot_image = rot_image.subsurface(rot_rect).copy()
    return rot_image
```