Final Project
Paper Asteroids

Philippe Lumpkin and Nicholas Pica

Abstract

Our purpose is to create an entertaining arcade game for our classic game lovers.Our goal is to create an Asteroids remake, that will follow a basic idea of the original game, but will have our own touch to it to make it stand out from the rest.

1. Introduction

For this project we plan to remake the well known game Asteroids. This 1979 game was developed by Lyle Rains, Ed Logg and Dominic Walsh. Asteroids was a major success, being one of the first of the golden age of arcade games. Atari, the game developers, sold over 70,000 arcade cabinets to arcade centers across the US. The game was designed using the Atari system and rendered on a vector 2-D display.

Our target audience would be those who enjoy a classic arcade game. The purpose of the project is simply to entertain the person playing. We hope to accomplish a remake of the base game along with our own tweaks. For example, an option to implement high scores stored in a database that the user can view.

1.1. Background

Both of our histories in this world have included gaming. We have always had a soft spot for them and always will. We began throwing ideas out for our project that began with a cookbook. That really wouldn't make since for us as neither one of us really even cook. Then the idea of making a game came along, which is way more in our ball park when compared to a cook book. Asteroids was one of our first thoughts and eventually became the project's concept. We figured that this is one of the top of the original arcade games and that it was simple enough that we could handle it, or so we though.

1.2. Challenges

We came across several challenges in this project. First, we have never created any sort of game in a c# wpf or any language for that matter. This meant that we were both going in very blind, and quite often we were underestimating tasks. Another challenge that arose was the generation of objects, like the asteroids, bullets, and spaceship. We attempted to do research online to gather ideas from other works but due to confusion we had to do the work ourselves. For this challenge, we solved it by adding the objects for the asteroids and the spaceship ourselves, and then the bullets are added into an array to be referenced via its element name. For the most part this works but makes the code longer than we would prefer. Another challenge, was the hit detection and this held us back for quite awhile. Eventually we stumbled upon some research that showed us how to create reference objects for each object on screen. With those we will run a function to detect whether the reference objects had intersected and if so then a "signal" was sent showing that it was hit.

2. Scope

We are hoping to create a full Asteroids remake. This means that there is a point system based on the amount of asteroids shot. There are a set amount of lives. Asteroids will appear from random points on the screen and move in a linear patterns. If the spaceship is hit by an asteroid then a life is lost and the spaceship is placed back in the center in the screen without the previous asteroids being cleared.

First we want to create a moving background to give a more pleasing look. We also want to make the asteroids break apart into smaller asteroids after destruction. Another possible goal would be to add more color to the game.

2.1. Requirements

We have a total of 8 requirements for our project. Most of our requirements were gathered from research of the original game and how it worked and played. We also based it off of how we have experienced similar concepts in other games

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	viewing high scores	Player	Easy	3
2	starting a new game	Player	Easy	1
3	adding new high score	Player	Easy	3
4	shoots asteroid	Player	Hard	2
5	player movement	Player	Medium	1

TABLE 1. USE CASES

aswell. Our first requirement is to have asteroids that appear randomly on screen and continue to move until destroyed. The next requirement is to implement a high score table that list the top 10 scores and that can be viewed by the user and updated when a higher score is made. This will be done in the same theme as the 1979 version entailed, displaying the top 10 scores and having a maximum of only 3 letters as the name. Our third requirement is to create a ship that moves and spins around the screen effortlessly. To install a working laser would be our forth requirement. The laser will be fired from the front of the ship and will damage/destroy hostile entities upon impact. The fifth requirement is to allow the ship, the laser beams, and hostile entities to exit from one side of the screen and appear on the adjacent side of the screen. This was also implemented in the original game by accident, however everyone liked in and so they kept it, so we intend to keep it to. The sixth and arguably the most important requirement would be to implement hit detection. This will be applied to the ship, the laser, and any hostile entities. The seventh requirement is to implement the breaking of the asteroids into smaller pieces. This will happen 2 times until finally destroyed as it was in the original game. The final requirement is to add custom audio files for the sounds effect in the game. We decided to add these custom files to bring a unique feel to the once futuristic sounding game.

2.1.1. Functional.

- The game will record high scores. The user will be prompted at the end of the game to create a gamer tag which will then be added to the high score database.
- The user will have 4 lives. When the game starts will, 4 lives will be added.
- The game will run when the user hits the start button
- When the user presses the high score button the screen will display the high score list.
- The game can be exited when the user presses the exit button.
- Asteroids that appear and move across the screen.
- A ship that can move and spin around and shoot and hostile entities.
- A laser that is shot from the front of the ship that destroys hostile entities upon impact.
- ships will break apart into smaller pieces after being shot. This will happen a total of 2 times and then they will be completely destroyed.

2.1.2. Non-Functional.

- The game will run smoothly without screen tearing.
- The game's high score database will have a capacity of 50 scores.
- The high score database will not be corrupted if he game crashes.

2.2. Use Cases

Use Case Number: 1
Use Case Name: viewing high scores
Description: A player wants to access the highscore screen. The player then clicks on the highscore button on the start screen. Then the database is accessed and then displayed for the user to view.

1) Actor can select the "Highscores" button.
2) Accesses the highscore database.
3) Displays the data from the database is a easy to read format

Termination Outcome: The database is is displayed for the user.

Use Case Number: 2
Use Case Name: starting a new game
Description: A player runs the program and is presented with the start screen. The player would then only have to click on the "start" button to begin the game. After clicking start the user is then shown the game screen where the game starts up.

1) Actor can press the start button
2) Runs the function that begins the game.

3) Displays the play screen.

Termination Outcome: The game begins for the user to enjoy.

Use Case Number: 3

Use Case Name: adding new high score

Description: A player finishes the game an is prompted to input their initials. After the player has input their initials then the screen switches to the highscores screen and displays the top 9 scores.

Use Case Number: 4

Use Case Name: shoots asteroid

Description: A player presses the fire button at an asteroid and hits the asteroid. This will cause the hitdetection funciton to be enacted to so that the asteroid is ”destroyed” and then it is sent thru the respawn funciton to be respawned. With this, the player is also rewarded points for each asteroid hit.

Use Case Number: 5

Use Case Name: player movement

Description: A player presses any of the movement buttons (WASD), and based on the button pressed a corresponding movement is applied. There is a function for ship movement that keeps checking for which button is pressed and when it is it will alter the ships position based on the movement. There is also another function that runs to check if the player has reached a border and if they have then this will cause the ship to appear on the other side, making the screen borders seem connected to the opposite side.

1) At the end of the game, the player is prompted to enter a 3 character name.
2) Accesses the Highscore database.
3) If the player scored enough to be in the top 9, then their score gets added to the database.
4) Displays the top 9 scores from the highscore database.

Termination Outcome: The user sees their new score on the screen.

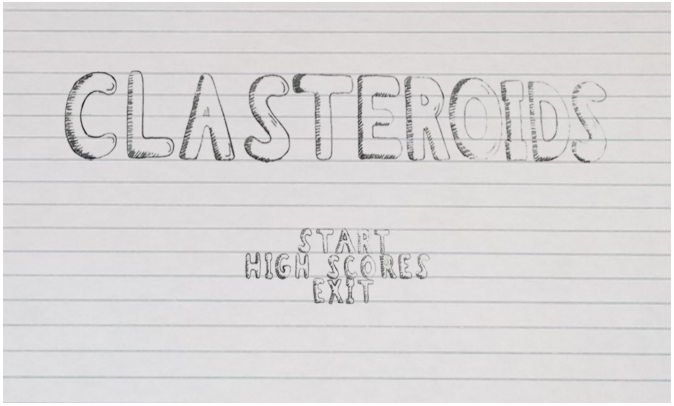

Figure 1. The is what the start screen looks like

Figure 2. The is what the game screen looks like

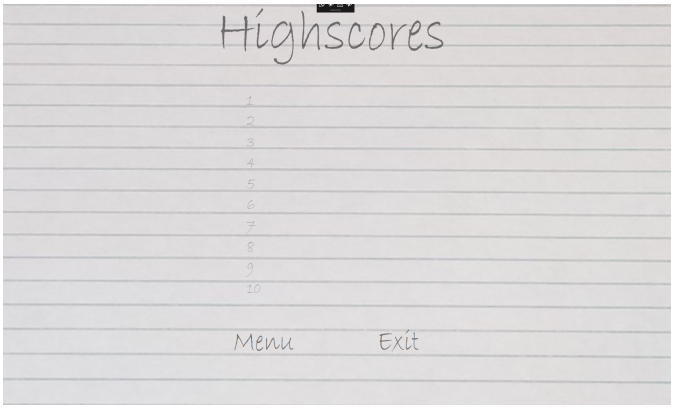

Figure 3. The is what the highscore screen looks like

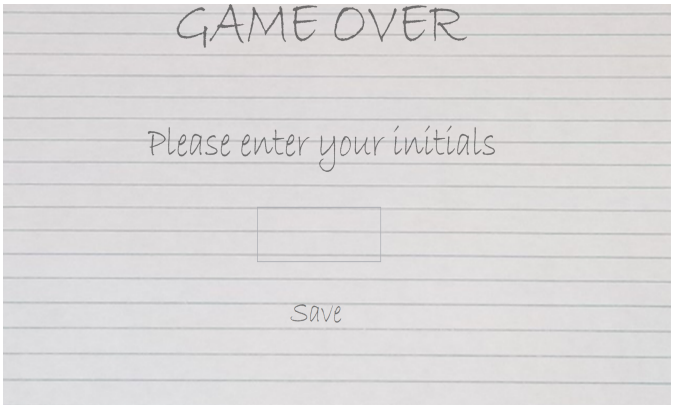

Figure 4. The is what the GameOver screen looks like

2.3. Interface Mockups

Figure 1 represents how we decided our starting screen to look like. This shows the title of the game and presents the player with three different buttons to press. Start will start the game and change the screen, highscores will show the highscore screen, and exit will exit the game. Figure 2 is our representation of the gamescreen. At the top left there is the score and the lives counter. There is also the ship and asteroid objects. The asteroid objects will be already moving on screen and the ship will be waiting for the player to move it. The last mock-up is the figure 3 which shows off our highscorescreen. This will have three column showing the position, initials and score of each player in the list. There are also two buttons at the bottom that allow the player to go back to the original menu screen or to exit the game. Our last screen is the Game Over screen, which displays ”Game Over” at the top and then asks the user to enter there initials which will then be added to the high scores screen. For example, see Figures above 1 (Start Screen), 2 (Game Screen), 3 (High Score Screen), or 4 (Game Over Screen).

3. Project Timeline

We started working on the project on March 15. From the 15th until the 21st we worked on the bare-bones of the project. We roughly implemented the asteroids that move from one side of the screen to the other. We then implemented a access database to display the high scores which we later changed to a simple file called highscores.txt. The last thing we were able to create in this time frame was a simple ship that would move and rotate across the canvas. We then didn't get a chance to work on the project until April 12th when we hit the ground running. From April 12th until the 22nd we were able to complete 3 more of our requirements. We created an a boundary around the edge of the screen that would respawn the asteroids back onto the screen on the adjacent side. This would give the effect of a continuous screen that would just wrap around. We then implemented the hit detection for the ship, the asteroids. After the hit detection we were able to implement lasers that used the hit detection and would blow-up the asteroids upon impact.

4. Project Structure

Our project is centered around a WPF window that once certain buttons are pressed or specified events occur then the window will alter its contents into a desired page. The Main window will open and start the ”StartScreen” page that includes the buttons Start, Highscores, and Exit. Based on the button press the buttons will either lead to the ”GameScreen” page, ”HighScoresScreen” page, or exit the entire window that the project is hosted inside. This was done so that navigation was pretty straight forward and seamless. We found that this way prevented us from having to open and close windows when a new page was needed. We have created separate classes for both the Asteroids and the Spaceship to make it easier to understand where all of the information is stored for the program. Our project incorporates a small database to store the information for the high scores.

4.1. UML Outline

So the outline starts at with a window of which all of the class exist with in as different pages. This was so that we could navigate amongst everthing fluidly. The main window immediately leads to the start screen. From there the user can access the Game screen or the Highscore screen. The highscore screen can return back to the start screen. The gamescreen is where a majority of the project is located. From here we have all of our objects such as the asteroids, spaceship and the bullets. On the gamescreen there is a state checker that will allow the game to paused and resumed if the player so desires. Here is the UML outline for the Asteroids project: 6.

4.2. Design Patterns Used

The first pattern we decided to specifically implement was the Observer Pattern. The idea is that there will be a constant observer ran over the spaceship that will watch for the isShipDestroyed bool to come back true. When that occurs, the observer will then update the GameScreen lives to one less. The second design pattern we are using is the State pattern. We implemented this in a way that the game could be paused and resumed by the user. The code would determine whether the specific key was pressed and if it was then the games objects and timers would be paused.

5. Results

Currently we have our 3 out of 8 requirements completely finished. Our project currently has a high scores table which in implemented through access. We also have asteroids that appear randomly on screen and move to the other side off the screen, however we do not yet have hit detection or the feature that allows objects to move of the screen and appear back on the adjacent side. We also have a ship that moves and spins effortlessly within the screen. We are working now on the 8th, 4th and 6th this week and plan to work on the last two (the 7th and 5th) from April 18th-24th.

5.1. Future Work

There are still some bugs that are currently in the build. Obviously we would love to fix those. This took a lot of hard work and dedication from the both of us, so definetly would not want to burn the project up. A good idea would to be finalize the project and have it on our GitHub so that we can have it either as a reference or for others to see what we accomplished. It also wouldn’t be a bad idea to rebuild this project in a better engine for gaming. While a wpf was fun to work with, there is software that can make this project a bit easier to complete.

Requirements	Design	Implement	Verification	Maintenance
1) Allow asteroids to appear randomly on screen, moving across to the other side.	We plan to have several asteroids appear in random locations at random intervals moving across the game screen.	Currently we have 5 asteroids appearing randomly and moving across the screen.	We have achieved what we have set out to do with this requirement. However, we may choose to implement more than just 5 asteroids.	This is working just how it was expected to run.
2) To implement a high score table using an Access database.	We plan to create a database with a very similar look and feel as the high score table looked in the original game.	We have implemented a dummy database for the moment so we can see how it works and what it will look like when finished.	The dummy table looks similar to the way it did in the original. We might, however, find a better font and change the size of the text to fit the screen better.	The database is connected and displays the information as expected.
3) To create a ship that moves and spins effortlessly throughout the game screen.	We plan to create a ship based on a lined-paper theme.	We have currently implemented a ship that moves and spins, we now just need to add the jpg file to make it have the lined-paper look.	The ship is implemented as planned, except for the theme.	The ship interacts with the game screen as intended.
4) To implement a working laser that works well with hit detection.	We plan to create a custom laser following the lined-paper look.	We implemented little black rectangles that act as our lasers.	The lasers work as intended and they fire from the front of the ship.	They work just as planned other than having them go along with the lined-paper look.
5) Allow asteroids and the player to be able to go off one side of the screen and appear on the adjacent side.	We plan to have this work just as it worked in the original game.	We set a border using x axis and the y axis. When the asteroids would interact with the boundary it would respawn them on the adjacent side, giving the effect of a continuous movement.	The asteroids interact with the boundary as intended.	Everything runs effortlessly.
6) To create hit detection for the asteroids, the ship, and the lasers.	This will be a very simple hit-box implementation. When the laser hits one of these items it will blow up.	We have added hit detection between the asteroids, the screen boundary, the ship and the lasers.	The asteroids disappear when hit by a laser and the ship is destroyed and respawned in the center of the screen once hit with an asteroid.	The hit detection works instantly with no errors.
7) Add exploding effects to the asteroids and to the ship.	To make the explosion more visually appealing rather than just popping out of existence.			
8) Create custom audio files.	We plan to record own voices and upload them to the game.			

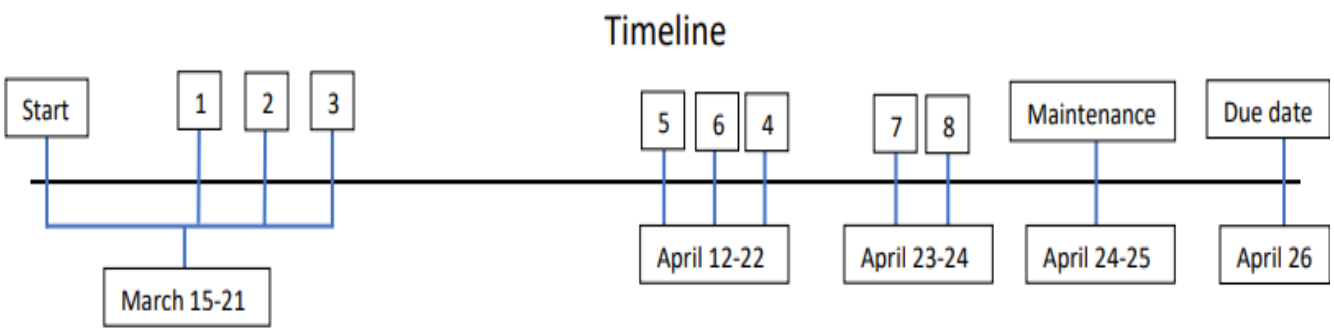

Figure 5. Our timeline for this project

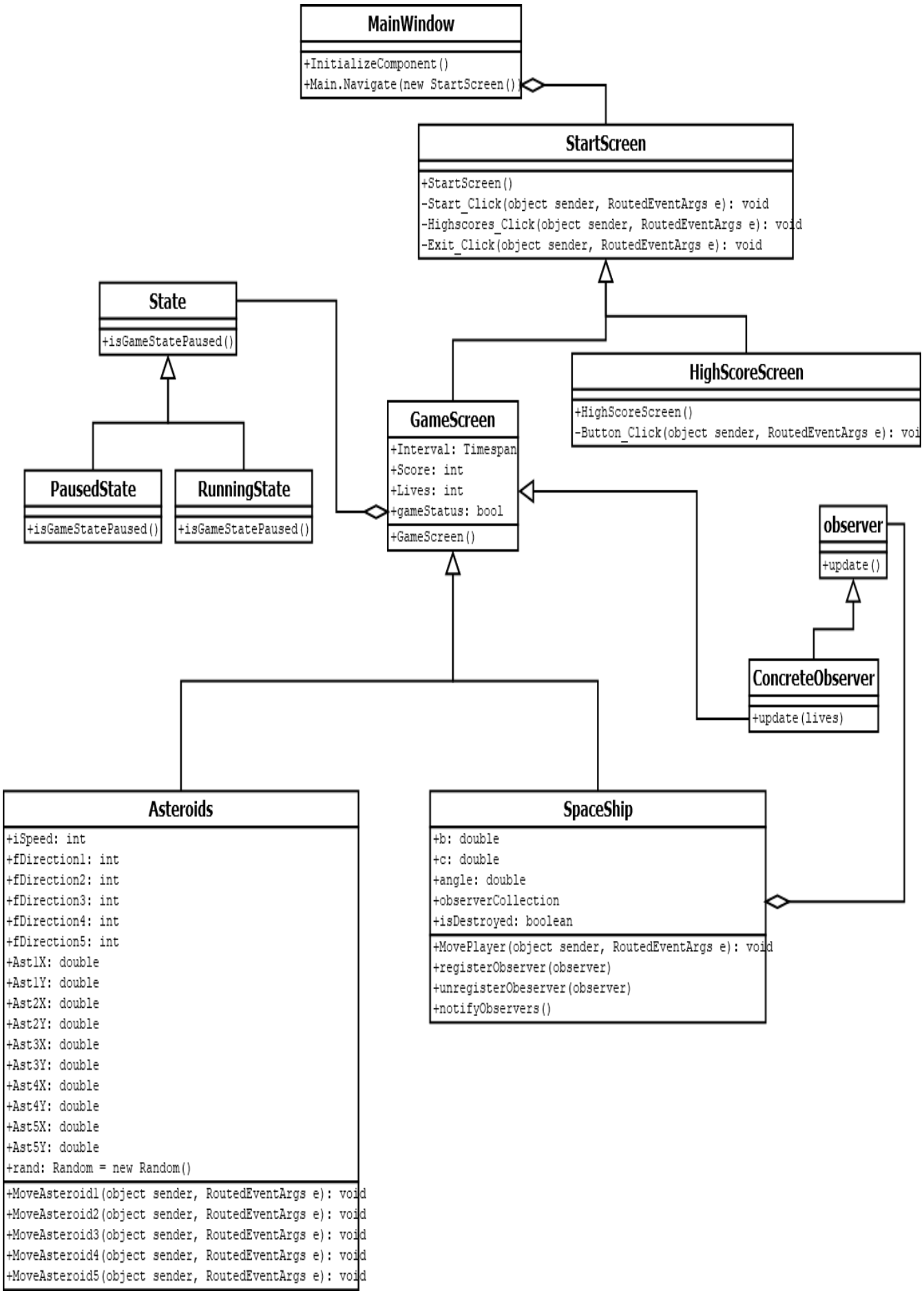

Figure 6. Our UML layout for Asteroids