University of Essex

School of Computer Science and Electronic Engineering

Course: Computer Science BSc

“Recreating a classic arcade game”

Asteroids

Student: Franciszek Kolebuk

Supervisor: David Richerby

Second Assessor: Richard Bartle

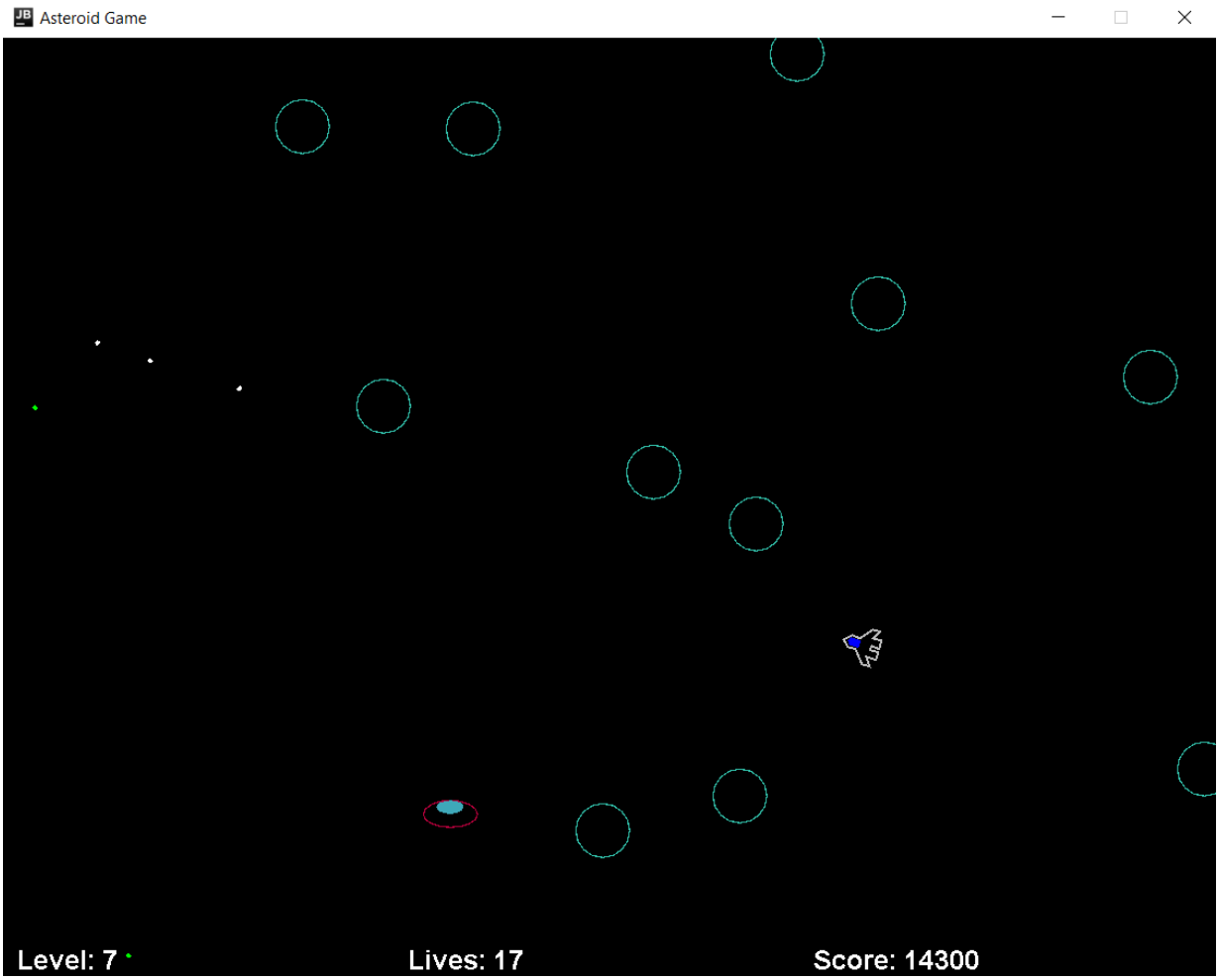

Figure 1 - preview of the game - source: self-made screenshot

Acknowledgements

I would like to acknowledge a some people, and thank them for their help.

Firstly I would like to thank all people involved in preparing materials and sources for the academic course CE218 Computer Game Programming, and *Dr Michael Sanderson* who was lecturing on this subject. The knowledge I gained during the classes in this subject turned out to be extremely useful during this project. I would also like to thank all people responsible for the course CE203 Application Programming. Materials and sources prepared by them also gave me some knowledge useful for this project.

I would also like to thank miss *Julia Fąściszewska* for spending her time on teaching me better usage of program MS Word and its many useful functions, which was essential for me to write this report.

The entity I would like to also acknowledge and thank for is the community of "Stack Overflow"[1]. During my works this year many times I needed to find something about Java syntax, or to understand working and implementations of some build-in functions and many times I could find answers for my questions there.

The same is true for online portal GeeksForGeeks[2] which also was quite often a good source of answers for a quick questions about coding in Java.

Last, but not least I must warmly thank my project supervisor *David Richerby*, who was guiding me for the whole this year. I have received many useful advices from him, and he also was extremely understanding and supportive, especially when I had a worse period and was dealing with some private issues that was affecting my works negatively.

[1] Stack Overflow - https://stackoverflow.com/
[2] Geeks for Geeks - https://www.geeksforgeeks.org/

Abstract/Summary

The main goal of a project was to choose one of the classic arcade games, and recreate it from scratch, without using gaming-orientated graphics libraries. I have decided to recreate a classic Asteroids from 1979 in Java. I also came up with an idea to, if time allows, go further with the original idea and after finishing recreating original Asteroids I wanted to create a second version of this game, based on the first one, which I could try to improve, add more features, and modernize its look. In the end I achieved to create a game, that can be played, and has most of needed features. Player controls a spaceship and can shoot bullets. Asteroids are appearing on the screen and moving randomly. All objects have implemented collision detection, so asteroids can hurt player's spaceship by colliding with it, and bullets can destroy asteroids. There is also a score counting system implemented, and player's lives that are decreasing when player is hurt and increasing when right amount of point is gained by him. After destroying fixed number of asteroids on every level player moves to another one with more enemies. All objects are appearing on the opposite site of the screen if gone out of it completely. Finally I added flying saucer, a new kind of enemy that can as a player’s spaceship can shoot bullets. After finishing first version of this game, I’m willing to create a second, much more improved one, add a lot of new features and modern looking graphics to it. I would like to find out if such great, popular game as Asteroids can be developed without making it actually worse.

Table of contents

Glossary

“Space Invaders” – classic “shoot ‘em up” arcade game from the 1978.

“Dying” – object being harmed by other object which leads to its removal from the game.

“Losing” – not meeting the requirements to keep playing, for example losing all lives.

GUI – Graphical User Interface

“Gameloop” – it is a loop of which the updating of the game is based on. It allows game to be run.

“Flying” – means various game objects moving across the screen. As game has a space theme every object movement is simulating flying, or drifting in space.

“Level” – is separated a stage of the game on which there are particular enemies and goals to be achieved.

“Destroyed” – meeting requirements to be removed from the game, for example getting shot.

“Shoot” – cast a moving projectile from the object.

“Teleportation” – disappearing and then immediate reappearing in a random place performed by the player’s spaceship.

“Warping” – inspired by the science-fiction classic movie series “Star Trek” term describing teleportation of the spaceship.

“Indie game” – game created by the game developing studio independent from any major game publishers.

1. Context of the project

.1.1. Classic arcade games

Arcade game is a specific type of a video game running typically on a machine made entirely for that purpose. The arcade games machines was usually located in an easy accessible place, mostly in a popular public businesses including but not limited to various kinds of restaurants, malls, pubs, amusement parks and most associated with them arcade games rooms. Machines was operated by coins inserted into the machine by a user, which allowed them to play an arcade game installed on a machine usually for a certain amount of time, or until “losing”, which means breaking one or more of predetermined rules of the game, such as losing all point, or simply “dying”. The purpose of this kind of games and machines they were running on was simple and it was to earn money for their owner. Playing arcade games required usually some level of skill and practice to be good at them, which would force users to play more, and more and hence insert more, and more coins into the machine, if they wanted to achieve high score. Arcade games even encouraged players to compete, by showing them lists of the highest scores got by others, promising that if they would play enough and practice their gaming skills they would also be able to get such achievement, or even greater.

.1.2. Introduction and history of Asteroids(1979)

One of the so-called classic arcade games, as well as the game I choose to recreate for the purpose of this Individual Capstone Project Challenge was “Asteroids”, created by Lyle Rains and Ed Logg. “Asteroids” was a released in 1979 by Atari, Inc. firstly on arcades and in following years it was ported to many various Atari consoles and computer systems. The game itself was a what can be called a multidirectional shooter, so unlike for instance “Space Invaders” player could shoot projectiles in multiple directions. In the game we play the role of a pilot who controls a little spaceship and can shoot bullets at the incoming asteroids and flying saucers, increasing his score in this way. The point of the game was to get the highest score possible by destroying all asteroids and occasionally flying saucers, at the same time not letting your ship to collide with any of them. With time and the successive levels, the level of challenge constantly grew due to increasing number of objects and enemies. The game was ending when a player lost all his “lives” which he could lose by colliding with any other object, or being shot by one of the flying saucers bullets.

The game can be called a classic arcade game due to their undoubtedly popularity in the period from the 1970s to 1980s with many arcade machines and home systems sold.

.1.3. Sequels and impact of the gaming industry

As the popularity of the "Asteroids" was increasing the time had come for its sequels, imitators and projects heavily inspired by it. The direct sequel to the work of Rains and Logg, by the name "Asteroids Deluxe" was released by the same developer Atari, Inc in 1981 on arcades, and in the following years there were made ports for various different platforms. Sequel was mainly the same game as the original, but was fixing one big commonly used exploit, changing some features like players ability to appear and disappear in random places across the screen to consumable and rechargeable energy shields, and was also introducing new kind of enemies. Nothing that would be change entirely the main core of the original "Asteroids" gameplay, but enough to develop and improve the idea of its precursor.

2. Structure of Asteroids and its features

.2.1. The main idea and point of the game

"Asteroids" allowed you to feel as the space pilot flying across the universe encountering new dangers on his way. The point of the game was to get as many points as possible establishing the highest possible score. Player could lose the game by simply colliding with any other object, like asteroids, flying saucers or enemy bullets. Game was introducing also a list of various interesting features that would diversify the game and make it more interesting .

.2.2. All features

"Asteroids" can be seen today as a quite a simple piece of software, especially comparing to modern AAA games[3], but it had an impressive number of features for its time including: controllable by player spaceship, randomly moving asteroids, additional type of enemy in the form of flying saucers, bullets shot by both player and flying saucers, useful but risky hyperspace, wraparound, various sounds, constantly increasing score and precious lives. In next section I am going to discuss all of them one by one.

.2.3. Spaceship and bullets

Player was in control of movement and also rotation of a small triangular-like shape which was supposed to imitate a real spaceship. By pressing the corresponding button player was able to move his ship in a direction pointed by its prow. The ship was moving at the steady speed as long as player was keeping the button pressed, and when released it would slowly start losing its speed to eventually completely

[3] AAA games – term from the video game industry describing games produced by the biggest companies with the highest development and marketing budgets.

stop moving. Two other buttons allowed to rotate the ship clockwise, or counter clockwise. The longer the kept them pressed the more they would rotate his ship. It is important to mention that spaceship was inert, which means that rotating it would not change the direction of it current movement, but only the direction at which it was pointing. To start flying in that direction player would have to press button responsible for forward movement once again.

Another crucial action that the player can take was shooting bullets from the prow of his spaceship. By rapidly clicking he could release certain amount of small projectiles in the direction his ship was pointing at. Bullets would fly for a fixed amount of time and then disappear, unless not collided with some asteroid or flying saucer.

.2.4. Asteroids

One of the core parts of the game was multiple objects called asteroids. They were appearing on the screen, close to the borders at the beginning of each level and flying in one random direction without changing neither it or their speed, until they were hit by one of the players bullets, which depending on their size was either removing them from the game completely, or transforming into two smaller asteroids flying in more or less opposite directions at higher speed that larger ones. The large and medium asteroids were splitting into smaller ones, and the smallest were destroyed when hit. Asteroids were the main enemy of the game, and destroying all of them on a level allowed user to proceed to the next one that included more asteroids to avoid and shoot. The asteroids themselves could harm the player's ship by colliding with it, so that would destroy both the asteroid and the player's spaceship. They could not collide with each other, or the flying saucers.

.2.5. Saucers

The second type of player's enemy in "Asteroids" were flying saucers. They were appearing once in a while on a screen, coming from the outside of the board in the form of UFO like moving objects, which could shoot some bullets that were being able to destroy player's spaceship when hit. Unlike asteroids they were not flying all the time in the same, randomly predetermined direction, but were changing it slightly once in a while, which was making them much more dangerous type of enemy. The game included two different types of the flying saucers, one bigger, one smaller. The bigger saucer were much less dangerous, because its larger size was making it an easier target, but also because it was shooting its bullets in random directions, without endangering the player's ship too often. On the other hand the smaller saucer was the biggest danger in the game, because it was harder to hit it and it was shooting frequently and accurately at the spaceship.

.2.6. Hyperspace

Besides flying around and shooting player could also take one more important action, which was called hyperspace. By clicking a corresponding button he could make his spaceship immediately disappear in one place and appear in another. The place where player's spaceship could appear was every time random and it is also worth mentioning that there was no protection that would stop it from appearing on, or really close to an asteroid, flying saucer or its bullet, which would most likely destroy player's ship right away. In that case the hyperspace ability was useful, but it was also really risky, because player could not know if his ship would not appear on the other dangerous object's path. Last but not least, the ability to hyperspace was also immediately stopping player's ship's movement, so after disappearing and appearing again it would have to regain the speed once more.

.2.7. Wraparound

Wraparound was a pretty simple, but extremely crucial feature as well in the original "Asteroids", as in all sequels and all similar games. It basically stops all kind of objects from going out of the screen borders. When a player's spaceship, or asteroid, or bullet goes beyond the border it makes it appear on the opposite side of screen. In that way nothing will disappear from the game. Wraparound was not changing the direction of an object, nor its speed, so it was allowing player to predict when for example the asteroid would appear after going beyond the border, so he could prepare his spaceship and shoot some bullets in that direction already. Some skilled and experienced player was using it as a one of their tactics for quickly destroying asteroids.

.2.8. Score and Lives

Two features that was essential for the game to even be playable and have any sense were score and lives. Score was what was driving the player to continue playing, as well to give the game another, and another try. Player was getting some points every time after destroying an asteroid and also after shooting down any flying saucer. Player's score could go up until the highest score possible which in the original "Asteroids" was equal 99,990 points. For every 10,000 points payer was rewarded with an extra live. Lives he could lose by colliding with asteroids, flying saucers, or its bullets, which would destroy his ship and place it again in a starting position. The game was ending when player lost all his lives and if his score was high enough game was allowing him to sign in to the list of the highest scores.

.2.9. Sounds

As every video game "Asteroids" had some kind of sounds playing during the game. Most of the objects like player's spaceship, saucers or bullets would make some sounds constantly, but there were also sounds assigned to specific actions and situations. For instance using hyperspace would make a special sounds assigned to specific actions and situations, also getting a new live, or dying would cause a specific sound to be played.

3. Recreating the game

.3.1. Early works

Choosing the game and the environment

I have decided to choose "Asteroids" as the classic arcade game for this Individual Capstone Project Challenge – "Recreating the classic arcade game". From the beginning it seemed intriguing to me, but at the same time it appeared as an achievable goal. I code the most in Java and I also used to try to create some simple games in that environment just for fun, or as an assignments for other modules at the university, so using it for the purpose of this project seemed a good decision. Also a year before this project during one of the other modules I was learning how to create games in Java, so using it again now would be a great usage of the knowledge I learned from those classes.

List of acceptance criteria

At the beginning of recreating the game would be good to have a plan of work, and for that I needed the list of acceptance criteria, that would be telling me what exactly I need to achieve to in the end be able to call my game finished. I decided to split it into two parts, one was the list of requirements that had to be fulfilled to have a playable game, and the other would include all features needed for full recreation of the game. For a playable game I obviously needed some kind of GUI[4], to be able to display everything to the user, controllable spaceship that could both move and rotate, some kind of "gameloop" I could base the whole gameplay on, asteroids as the basic enemy to shoot, bullets, collision detection, so the bullets could actually destroy asteroids, and asteroids could harm player's spaceship, implementation of deaths, lives, points and last, but not least some kind of score systems, and changing levels. Making all of the listed would allow me to have playable game. To make a full recreation of the game I would also need some additional features including: implementation of hyperspace ability, asteroids splitting into smaller ones, sounds played in appropriate moments, and saucers flying and shooting at player's spaceship. After listing all acceptance criteria I could start my work.

[4] GUI – graphical user interface

Preparing Graphical User Interface (GUI) and creating board

First and the most basic task for me to do was implementing GUI and creating some kind of board that everything would appear on. As I used it many times before I decided to use build-in JComponent library and JFrame to create a visible frame, with a fixed size. I tested a bit different colours, but due to trying to recreate a specific game I set the background colour to black, as in original "Asteroids". To have everything visible, and being able to repaint it many times in the future as well as being able to create geometric shapes easily I used to Graphics2D class from java.awt build-in package. When this was finished I was able to move on to creating the spaceship controllable by player.

Player's ship

Before creating a ship itself I needed to have a GameObject interface for all my future objects. I planned to add there later all functions that could be applied to objects in my game, like for example being able to collide with another object. As all of objects in my game would be moving they would have their current position, speed and size. For that I used a Vector2D class, that was introduced to me and another students during my second year of education at university. With that I was ready to create a class for my ship that would extend GameObject interface. I used a Graphics2D method drawPolygon to draw my spaceship, and at first I drew a simple triangle like in the original game, but I wanted to do something better looking, so I spend some time creating a shape of a ship that would satisfy me. I set his position to be exactly in the centre of the screen, and in this way I had a spaceship for a player, but it was not moving yet.

Figure 2 - Spaceship - source: self-made screenshot

Vector based movement

I had a ship but I could not move it yet. I needed to create a movement system for it. As I implemented earlier vectors with a Vector2D class I was able to use it for this task. I set the maximum acceleration for my ship, I wanted it to slowly start gaining speed when a corresponding button would be pressed by player and fly forward as long as the player would keep this button pressed. At the same time I created class Action that would handle all actions I could apply later to my ship. Later I was updating it every time when I was adding another action for player's spaceship. My ship was also supposed to turn left and right, so I made him rotate by a certain angle multiplied by time the player was making it turn for. Later during the first test it appeared that the ship was a little too responsive making precise aiming with it hard, so I added a very small delay time which fixed it. I knew that in the original "Asteroids"

spaceship had a certain inertia, I explained how it worked in details in chapter 1.3 “Spaceship and bullets”, but honestly it was a one feature from the original game that I was never a fan of. In my opinion in made game a little to chaotic, and irritating. I was supposed to spend months with similar game, running it and testing many, many times and, often needing to check a feature quickly and efficiently and an inertia that would make controlling the ship imprecise and chaotic was not a good idea in my point of view, so quite early I decided to not add that functionality to my game. At the end I still think it was a good idea.

Spaceship controls

I implemented movement, but I did not link it to any buttons, mouse, or anything. It was time to make class Keys that would handle controls of the player’s spaceship. Before that however I needed a little public interface Controller that would be implemented to this class, so I could connect it with actions I created in class Action. My class Keys extends KeyAdapter and KeyEvent build-in java libraries. I prepared in that class many switches and its cases for every button I wanted to use. Added also a simple long variable that would calculate the time a key was pressed for. My game was supposed to be play on personal computers, so I used keys from a simple computer keyboard to be buttons responsible for the movement of my ship. In these times players usually use arrow keys, or “W”, “A”, “S”, “D” key group to play games, besides I wanted my game to be adapted to both right and left-handed players, so I made able for player so use any of those two options. “Up” and “W” keys allowed to move ship forward, “left” and “A” turn it left, “right” and “D” turn it right, and “down” and “S” reduce the speed accordingly. The longer the particular button was pressed the longer the action it was responsible for was being executed. All essential controls for that moment were complete.

.3.2. Playable game

Implementation of “gameloop”

After finishing early works stage and implementation of spaceship controls I had everything needed to have a ship that could be controllable by player, but I needed one, extremely important thing that would made my game actually constantly running and refreshing, so all other object could move. I needed a “gameloop”, a simple loop that would run constantly, and stop only when the game would be over. I created it in a Game class, and made it update my game and repaint its view 60 times per second. In that way screen of my game would be repainted constantly, so player could see all objects that are not his ship moving.

Wraparound

Wraparound is a feature crucial for the whole game. It makes sure that nor object would eventually fly beyond the border of the screen. Instead it should reappear exactly at the same position, but on the opposite side, which means object going beyond the right border should reappear at the same height, but on the left border, and object going beyond down border should appear on appear at the same width on the up border. Every object should keep their direction and current speed. Creating that was quite an easy task, all I had to do was set some borders to the frame, and change one of the two position values to minimal, or maximal possible value every time an object was crossing the border. This kept every active object from drifting beyond the border endlessly.

Creating asteroids

The most basic enemy in the game was asteroids. In my version of the game they are just a big, single colour empty circles. I created a simple class Asteroids, which extends GameObject interface, for them and set their speed and size. I once again used Graphics2D to draw them. I tested a few colours, but I really liked the cyan like colour, so I just left it that was. After that in the Game class I made them appear in the fixed quantity on the screen every time the game starts. Starting point, their direction, and speed are every time generated randomly, and they do not change until destruction of them.

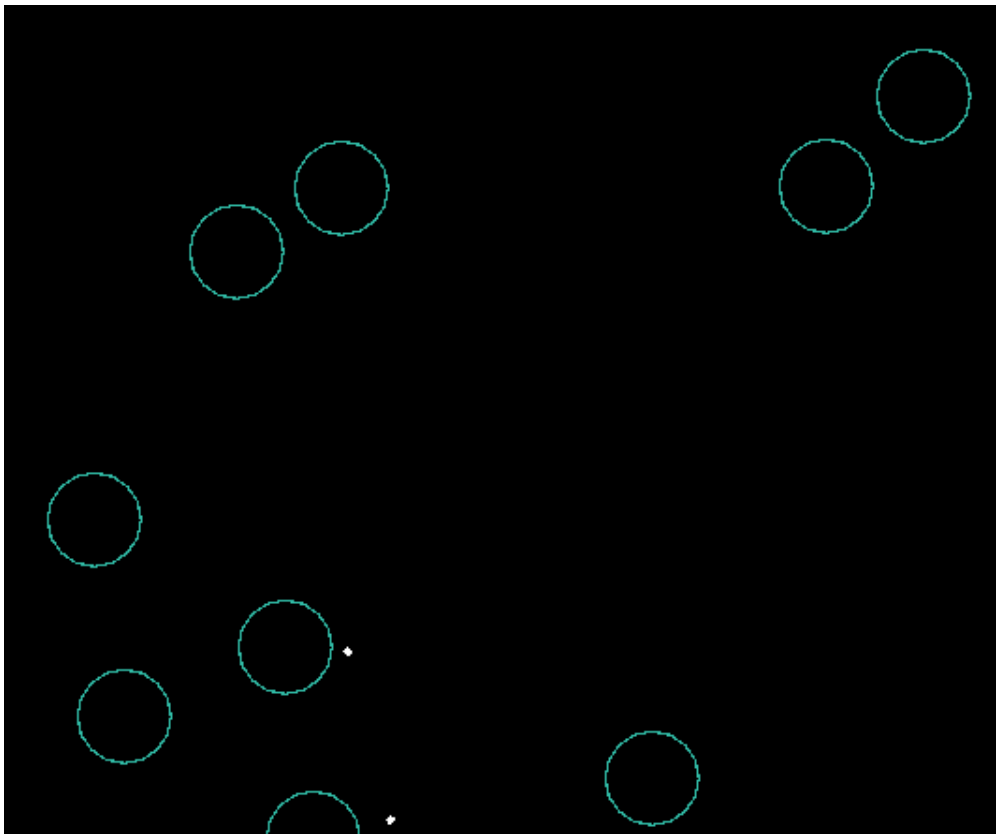

Figure 3 - Asteroids - source: self-made screenshot

Bullets

After making asteroid the game needed something that player's spaceship could shoot at them. That was the time to create bullets. Bullets was supposed to be small simple objects coming out of the front of the ship when player would press a corresponding button. I created a class Bullet that extends GameObject interface and as usual set their speed and size. Graphics2D once again served me to draw an object. The important change that would differ bullets from other objects was that they should disappear automatically after flying a certain distance from the ship. To do it I set their time to exist and when it was over the bullet was removed from the game. When created bullets needed to be linked with some action and some button that would invoke it. In class Action I added new variable shoot, linked it to the player's spaceship and created a new case in Keys class that would make a new bullet fly from the front of the ship every time the corresponding button was pressed. To not make game to easy and pointless I made some breaks between each shot, preventing the board from being flooded with bullets, by a rapidly clicking player. It would destroy the point of the game if the player could do that.

Collision detection

Having both bullets and asteroids is a one thing, but without properly implemented collision detection nothing happens when they meet. There needs to be some kind of system that would check if two objects of a certain type are overlapping, so there a specific action could be performed. Because I was not expecting too many objects in my game at the same time I decided to use probably the most simple way of collision detection, which was constantly checking the positions of every single object and calculating the distance between them and all other objects of the type they could collide with. If the distance between two objects that could collide were smaller than the sum of both of their radiuses then it meant that this two objects were overlapping, so they hit each other, and an appropriate action can be taken. To implement it I created a method overlap() in GameObject class, so it would be checking if objects are colliding. I also made method collisionHandling() which, if objects do not belong to the same class, sets theirs statuses to being hit. Unfortunately, nothing changes then yet.

This task was really important for the game to be playable, but unfortunately I had problems to make it work correctly for a much longer time than I assumed at the beginning. In the end it turned out that the syntax was supposed to be different than expected. To conclude, when it finally started working I was really, really happy and satisfied.

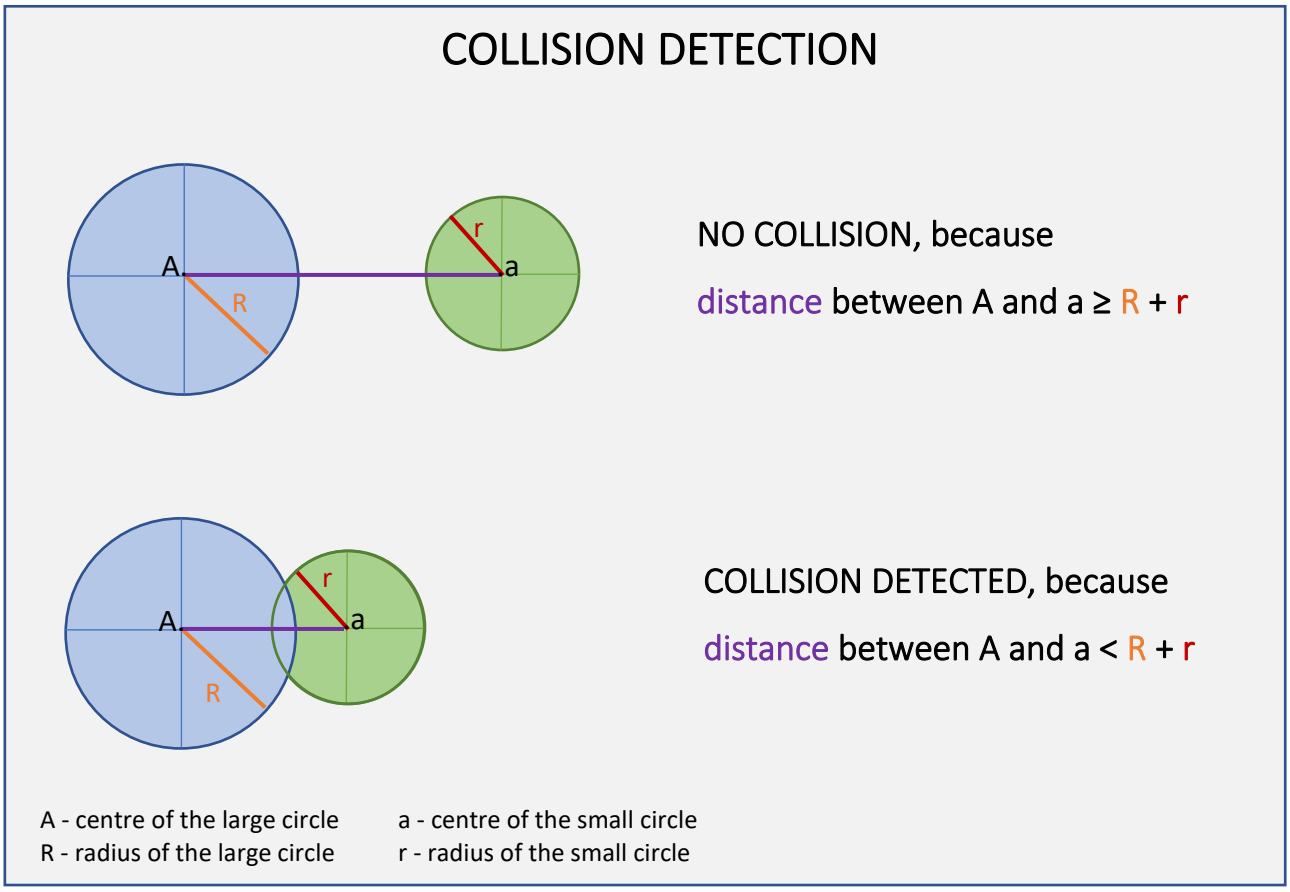

Figure 4 - Collision Detection - source: self-made

Death

When two objects collide with each other they both should be destroyed immediately. Collision detection is responsible for checking if they are overlapping and, setting their statuses to being hit, but it does not remove them for the game. For that I need a death status, that would simply remove object with this status from the game. I created a Boolean variable "dead" in a GameObject interface, so it could be applied to every single object. I set its default value to false and would change it to true only when particular object gets hit status. After that I created a condition in the Game class for every type of object, that states if the object's variable "dead" is true game is not updating it anymore, which simply means removing it from the game. In case of asteroids I also needed to decrease their quantity value every time any of them was destroyed. At this point of my work I ran into an extremely bothersome problem, that took a lot of time and my nerves. As far as I knew everything was supposed to work correctly, unfortunately instead of spaceship shooting bullets and destroying asteroids in this way I had a ship that was disappearing for some reason every time when the first bullet was shot. This was making any further game impossible, due to the lack of controllable by player spaceship. I spend a lot of time trying to understand what was the issue, I even considered changing completely the method of collision detection, but in the end I finally realized the problem was that the spaceship's bullets were appearing a

little too close to it, and collision detection immediately was counting it as a hit, so player's spaceship was destroyed. All I needed to do to fix it was changing a bit the position of bullet spawn relative to the spaceship. Someone could say that I should change the collision detection, so it would not check if bullets and player's spaceship collide at all, but I took some tests, and it appeared that due to bullets speed exceeding spaceship's speed there is no way that the ship could collide with its own bullet after shooting it. I even tried for some time to make player's spaceship hit its own bullet, but it was impossible. With fixed bullets, and death status implemented player could finally shoot asteroids and, if careless, be destroyed by them.

Lives and points

When the death was working correctly there came a time for lives to be added to the game. In the original game player was starting with the certain amount of lives, and was losing one every time his spaceship was destroyed. He could also gain a life once in a while if he met some predetermined requirements. To add lives system to my game I created some variables and methods in Game class that would set the initial amount of lives to the player's spaceship, check the current amount of its lives, and decrease, or increase its number in certain conditions. In my game player gets a new live for every 1000 points he would score and also for moving to the next level. Initially player's spaceship has 3 lives, and loses one when collides with any other dangerous object.

Important part of the original "Asteroids, and main goal of the whole game was getting points. The points were what was droving player to continue playing and therefore spend more money at the arcade machine. The structure of points was simple, player was getting a certain amount of points for each asteroid he destroyed, and each flying saucer he shot. The same as in live I just created a variable holding the current amount of points starting with zero, and a method that would increase it by 100 when any type of enemy was destroyed. I was also giving player 100 points for getting a new live.

Score system and levels

After creating lives and points I had to display them somehow, so the player could be aware of his current score and the lives left. Again with using appropriate for this task build-in library Graphics2D in View class I prepared some panels displayed at the bottom of the screen, and drew on them string with the current level, lives, and score. I linked it to the variables I created earlier, and from now on all these strings were constantly updated. With score and lives working it was time to make levels. I already had a place where I could display the current level, but not level system in general. In the original game player was moving on to the another level when all active asteroids and saucer on the previous one were destroyed, so I decided to do exactly the same. I created method reset in Game class, that would change the level every time when player moves to the another one. On each new level the number of asteroids was increased by the initial number of asteroid, which in my case was 5. After creating this part I finally had a playable game completed.

.3.3. Required features

Hyperspace

My game was running, and was playable, but that did not mean in the slightest that it was fully completed. There were still some important required features that were supposed to be implemented to it, and hyperspace was one of them.

The ability to hyperspace was supposed to allow user to teleport his spaceship in a random places at with. I created a new action "warp" that I applied to the spaceship, and I also prepared the shift key that would be responsible for this action. When occurred this action would change the current position of the ship to the random point on the board. In the original "Asteroids" it was also stopping the spaceship completely, but after a few tests I found this feature to be annoying and slowing down the gameplay, so I decided to just let the player's ship keep its speed and direction after teleportation. The worth thing mentioning is that as in the original game nothing stops player from teleporting randomly into an asteroid and destroy his ship in that way, or just reappear really close to the other object what could also end pretty badly for it. However, at the begging of the game is really hard for this to happen due to limited number of asteroids. It happens much often on the next levels, which is good, because the game is supposed to become harder and harder as the player advances to the next levels.

Sounds

Every good video game is supposed to play some sounds in a certain situations, so my game also needed that. For this task I created a special class SoundManager in a separate package "utilities". I also spend some time on looking for good quality sounds online that would be for free use, so I could use them in my game. Eventually I found page Freesound[5] that provides enormous amount of sounds for free use. I found appropriate for actions in my game sounds and put them in a specially prepared for that purpose folder in my project. Then I linked them to my code and prepared for later usage by using a few build-in Java libraries including *javax.sound.sampled.AudioInputStream, javax.sound.sampled.AudioSystem, javax.sound.sampled.Clip; and java.io.File;*. I created method playing chosen sound, and a few methods that would perform this method, each for different situation that required plying sound.

After finishing this part I tried to test it in the game and I quickly realized that in the game such dynamic as "Asteroids" there is a lot of things going on at the same time, and hence, a lot of sounds that are played at the same time, which sounds unbelievably annoying. I realized I implemented too much sounds, so to calm it down a little bit I had to delete some of the sounds playing. In the end I limited sounds in my game to playing when a bullet is shot, when player's spaceship teleports, when player dies, and a quiet constantly playing sound that plays when ship flies forward. In this way I added in my opinion the right amount of sounds, and implemented them correctly.

[5] https://freesound.org/

Saucers

Original “Asteroids” had more than one type of enemy to shoot in form of flying saucers. This was my next task – adding a new type of enemy. By the time I was trying to achieve that it turned out that it was much more complicated than I imagined at the beginning. The details were clear, create a flying saucer, that would appear once in a while, coming from beyond the screen border. It was supposed to fly in a random direction and occasionally shoot dangerous bullets at player’s spaceship. I started my work, and quite fast realized that probably would be better to start from creating the other type of the saucer, the one that shoots in a random directions. I created a class FlyingSaucer for this object. Because it was supposed to behave similarly to asteroids I made it really similar to Asteroid class. I made it look like a blue oval inside of empty red oval, and I was spawning it randomly at the beginning of each level. This part was fine to do, much harder was making this object shoot bullets. I tried and tested many ways, but eventually I realized that probably the best option was creating a separate class for saucer’s bullets. At the beginning I was planning to use the same bullets I was using for player’s spaceship, but recoloured. However, it was much better to have saucer’s bullets as a completely separate entity, because their behaviour and properties was different than the regular bullets and this was better and much clearer to code with different types of objects. Eventually I came up with an idea to cast saucer’s bullets exactly from the centre of the flying saucer, moving in random directions, but for that I needed to prevent saucer’s bullets from colliding with saucer.

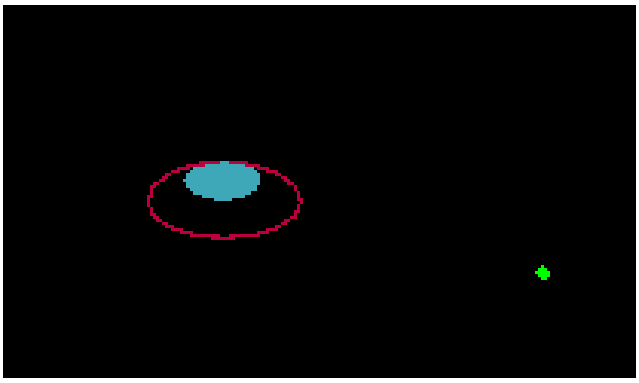

Figure 5 - Flying Saucer - source: self-made screenshot

Updating collision handling

At this moment I knew I needed to rewrite a bit collision handling, because I beside this problem I expected saucer’s bullets to collide with all other objects, including asteroid which flying saucer should not be able to shoot. My idea for collision handling was working at the beginning when I did not have many types of objects and various dependencies between them but now it was not doing its job at all, allowing every object hit every type of other object, no matter if they should be able to do is, or not. It was like that, because when checking if two objects collide with each other I was only checking if they

do not belong to the same class. Now I needed a new condition for it. It took me some time to apply it correctly but eventually I was able to fix everything, including ship's bullets being able to harm player's spaceship. Now bullets from player's spaceship can collide only with asteroids, or flying saucers, and the spaceship itself can collide with flying saucers, its bullets and also asteroids.
Unfortunately, at the moment when I write it the code still have some kind of bug, that stops it form letting saucer's bullets collide with anything, so flying saucers are moving around, can be shot and rammed, but their bullets make no harm to any other object. I do not know why it works like that, after finishing writing this report I intend to fix it as soon as possible.

Summary of achievements

By the time I have my final report finished I was able to achieve quite enough to be satisfied with my work. I created a playable game, with spaceship controllable by player, basic enemies in the form of randomly flying asteroids, bullets destroying them, sounds, ability to teleport, scoring system, lives, deaths and levels, and finally one other type of more complex enemies – randomly shooting flying saucers.

Testing

Due to nature of the project, it quickly became clear for me that usual Unit Testing would not work good here. Instead, during the whole process of the development of my game I was just simply testing it by running the game itself and observing the outcome, especially the part I was focused on lately. It allowed me to quickly find issues, and make be sure for all the time how my game would look like to the player.

4. The idea of improving classic game

.4.1. Is it possible to improve something that is already working?

At the beginning of the project, even when before I started writing any code I, during making my research for it I realized that original "Asteroids" from 1979, besides being a really good, and extremely popular in its time has some place for upgrade. I also recalled an "Asteroids Deluxe", direct sequel to the original game, that tried to fix some flaws of its predecessor and also introduce more new fun features, only 2 years after the first "Asteroids" were released. In my opinion the sequel was great, better than the first game, but its predecessor was much more popular. I started wondering if there is possible to improve something, that has achieved undoubted success and not make it actually much worse.

.4.2. The idea of modernizing classic game

Then I came up with an idea. Modernizing a classic game, trying to make it better without spoiling it. That sounded like an interesting and challenging task. Especially in the times when retro gaming is really popular, and every year new indie games with a retro theme are being released. I decided to consult it with my supervisor, he agreed with me that it is an interesting idea. If I could I wanted to add this as an additional task after recreating the classic game.

.4.3. Room for creativity

Unfortunately, due to various things and issues I was not able to realize my idea. In the given time I could only recreate the classic “Asteroids”, but I still think that the idea of creating a game similar to asteroids, but with fixing its flaws and implementation of the new features, would be something worth spending time on, especially because it would be a great room for creativity. Inspired game developer could realize all his ideas, add new things and maybe presented to the world something with similar, or even higher level of quality than the original “Asteroids” from 1979.

5. Future Works

.5.1. Splitting asteroids

At the end of each large, and the long lasting project there should be always plan for the future works. Due to me not being able to implement splitting to the asteroids yet doing it would be my first task for future works. It would probably take some more time and effort to do it correctly, mainly because of the need to completely redesign some parts of the code and its architecture, so it would be much more appropriate for this task, but I think this is an goal in my reach, just not yet.

.5.2. Creating an Asteroids Extended

Another example of the future work would be realizing my idea and creating a much more advanced, extended version of my game. It would include redesigning gameplay a little probably, adding new types of graphics, backgrounds, sprites and animations. Also would be great to have some kind of a music in this game, maybe different music themes on subsequent levels. Levels could also vary in terms of main theme and enemies. I also think that the new version of my game would introduce a lot of completely new features.

.5.3. New extra features

Creating new features would be a really great moment to show the creativity of a game developer. Time would probably show what is possible and what not for a single person to do, but I guess some new features could be pick-able power-ups, like consumable missiles, or rechargeable energy shields. I could also introduce much more types of enemies, with much more complex behaviour, maybe some bosses to beat at some levels. Introducing the online multiplayer mode, or split-screen co-operation mode would be also a great idea to realize. I certainly believe that new features I listed above would make this game much more interesting that the original “Asteroids” and I hope that someday I would be able to realize all of that ideas.

Project Planning

At the beginning of this academic year I was supposed to create a plan of working for this Individual Capstone Project Challenge. I planned to do as much of work as possible in the first months, because learned from previous years' experience I was expecting that the second term is usually more difficult and also at some point there can appear private issues unrelated with the project. In the end it turned out I was right assuming that, so I am really very relieved with my planning decision. Basically I wanted to have most of the tasks for the “playable game” part done by the end of Christmas, so in the next term I could fully focus on much more complicated and hard to implement features, like more enemies, or splitting asteroids. I am proud to tell that I was able to fulfil my own assumption quite good. Unfortunately, during second term I ran into some problems, and also I also a worse period in my private life during February and March. Details of that are irrelevant, but in general it had quite a negative effect on my motivation and progress in my work. Fortunately, near to an end of the project I was able to be productive again, and eventually still achieve something, but due to worse periods I was not able to create everything in the way I intended to at the beginning. For example I could not implement splitting asteroids, or even start working on my extended version of the game. In retrospect I think my methodology proofed right enough, thanks to good early start I was able to eventually complete this project, but it could be better prepared for issues, and worse periods in the middle of the yeah. I am satisfied with my achievements, a little less satisfied with my overall performance, but given the various conditions it was still pretty good. I learned a lot about game developing, managing a long-lasting project and a time developer can spend on it.

For the whole year I was using Jira to help me manage my task, and time. I present to you my Cumulative Flow Diagram which describes process of my works. Unfortunately it only shows progress when a task is completed, it does not show effort put into learning an research, but I still think it is worth mentioning.

I also add a link to my Jira below, however for some reason, probably security related someone at the University decided to let it be accessible only from university labs, or via Horizon[6].

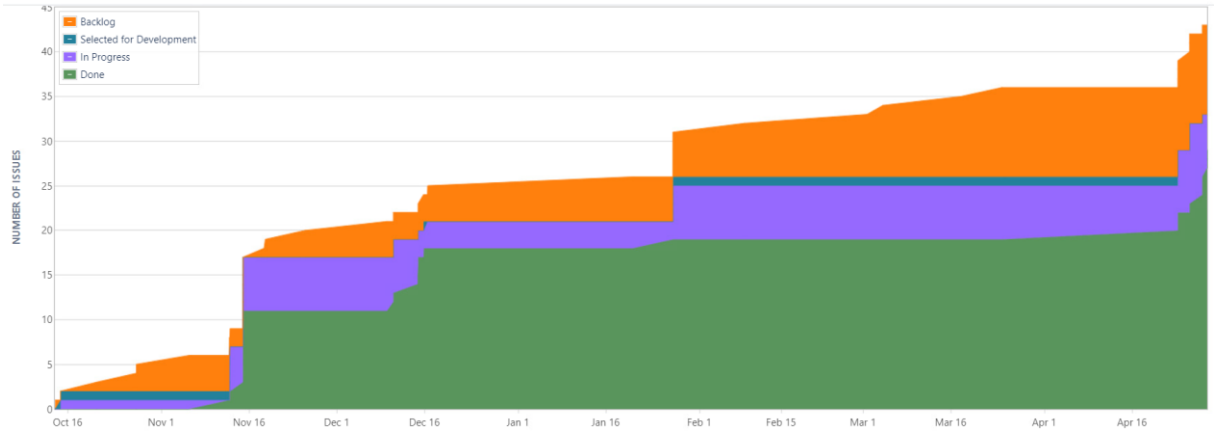

Figure 6 - Jira CFD - source: self-made screenshot

Jira: https://cseejira.essex.ac.uk/projects/B30167/summary

Code of my game can be seen in my GitLab repository to which links I include below. Unfortunately, at some point I did not realize that the project on my personal computer, opened in IntelliJ IDE was not correctly connected to my GitLab repository. Due to this fact for a long time I was making commits that was not pushed to the repository, but only kept in the Version System Control on my personal computer. Lately I realized this mistake, and I tried to fix it, but I ended up with creating a new project in my GitLab repository. I include both of that links below, the first one is the project where my code was supposed to be from the beginning. It includes my project at the stage from a few months ago. The other one leads to the project I created lately, and it has my up to date project, with everything needed, including commits history that was being kept on my personal computer earlier.

The first project https://cseegit.essex.ac.uk/ce301_21-22/CE301_kolebuk_franciszek_j
My up to date game: https://cseegit.essex.ac.uk/fk19730/AsteroidsGame

I plan to fix this in the nearest future, so everything would be in the right directory, but due to my current works on this report I was not yet able to do it.

[6] https://csee-horizon.essex.ac.uk/

Conclusions

To conclude, I think this project was really interesting, and challenging to make. I learned a lot of new things, including coding, task and time management as well as dealing with problems. I have spent half a year trying to do my best in recreating a classic arcade game, and I am really satisfied with the outcome, although I agree that maybe if not some given conditions I could have done a little bit better. At the end I was able to nearly fully recreate “Asteroids” from the 1979, meet most of my initial requirements for full completion. My game is playable, has more than one type of enemy, player can destroy asteroids with bullets, get points, lose and get lives and move to the next level after finishing the previous one. I was even able to implement some sounds to my game. Unfortunately, I was not able to make a few things yet. My game still needs to have better implementation of flying saucers, because there are some bugs with them. Also feature making asteroids split after hit waits for its time to be created. In terms of future works I need to make those two tasks I mention above, as well as create an extended version of my game. I cover all details about it in the two sections of my main text – “An idea of improving a classic arcade game” and “Future works”.

References

"Production Numbers" (PDF). Atari. 1999. Archived (PDF) from the original on January 20, 2013. Retrieved March 19, 2012.

Monfort, Nick & Bogost, Ian (2009). Racing the Beam. MIT Press.

"The Making of Defender". Retro Gamer. No. 55. Imagine Publishing. October 2008. pp. 34–39.

Chris Kohler (November 17, 2011). "Asteroids Designer Ed Logg Honored With Pioneer Award". Wired. Condé Nast Publications. Archived from the original on December 8, 2013. Retrieved December 28, 2013.

"The Making of Asteroids" (PDF). Retro Gamer. No. 68. Imagine Publishing. 2009. Archived from the original (PDF) on December 19, 2013. Retrieved December 18, 2013.

Salen, Katie & Zimmerman, Eric (2004). Rules of Play: Game Design Fundamentals. MIT Press. ISBN 0-262-24045-9.

Edge Staff. "The Making of Asteroids". Edge. No. 117. Future plc. Archived from the original on January 4, 2014. Retrieved January 4, 2014.

"What arcade games looked like before video games, 1968 - Rare Historical Photos". rarehistoricalphotos.com/. 20 August 2019. Retrieved 8 March 2022.

"SEC Info - Atari Inc".

"The Asteroids 'Lurker'". Electronic Gaming Monthly. No. 103. Ziff Davis. February 1998. p. 91.

"Asteroids Deluxe". Arcade History.

Sullivan, George (1982). How to Win at Video Games: A Complete Guide. Scholastic Book Services. pp. 81–82. ISBN 978-0-590-32630-8. In fact, in the first few months after Asteroids Deluxe was introduced, it proved so difficult that many players turned their backs on the machine.

Staff Writer (May 10, 1981). "'Asteroids game': trying to stay alive". Poughkeepsie Journal. Poughkeepsie, New York.