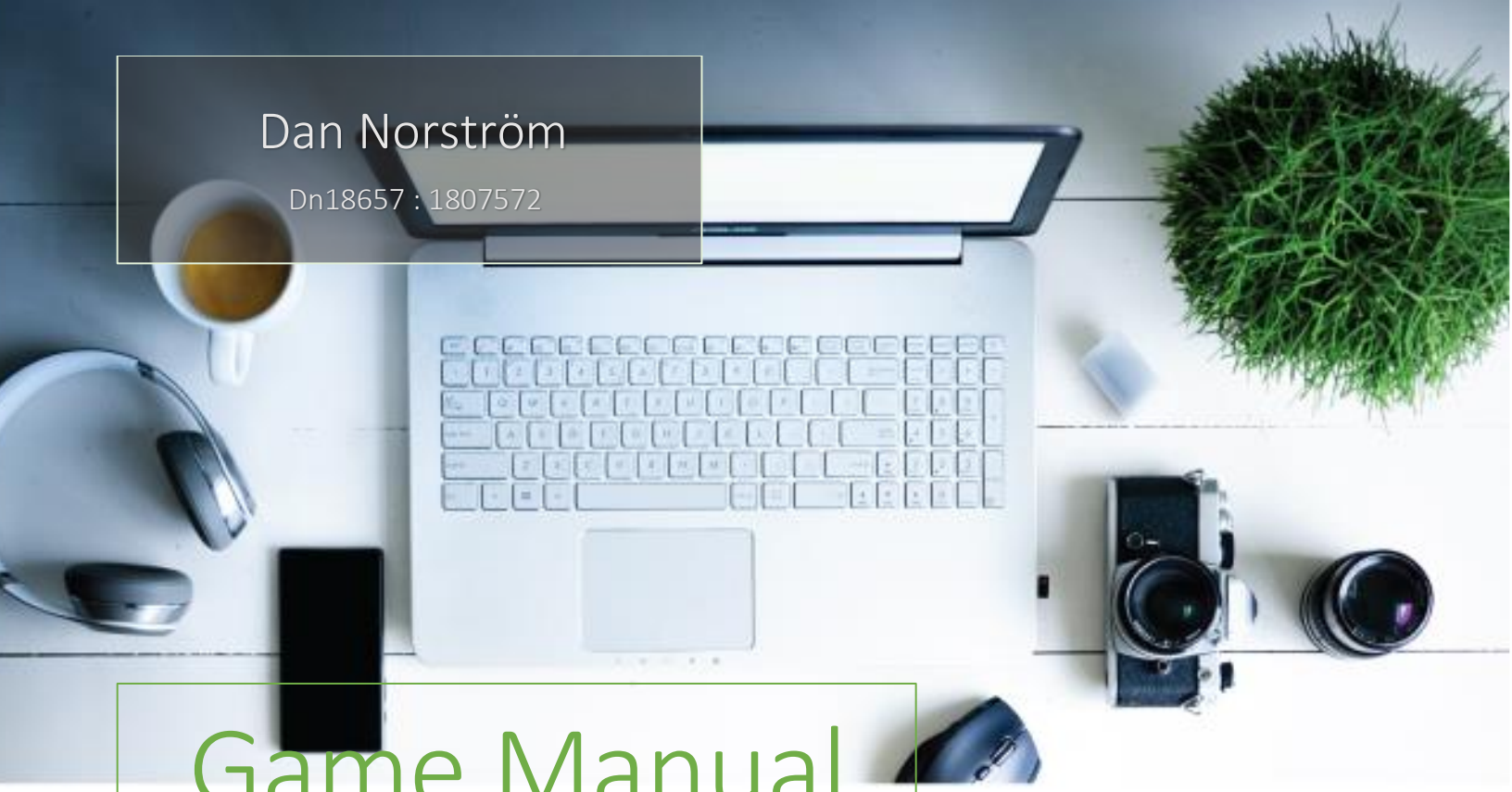


Dan Norström

Dn18657 : 1807572

Game Manual

CE218-5-SP : Assignment

The Story so far…

In a galaxy very close to your monitor, a single pilot challenges the six great empires of big floaty rocks while being pursued by the evil sandy spinny bois. Can he find some definitely, most positively trustworthy drones to help him on his journey among the salvaged parts of the spinny bois? Most likely not, but the legends tell that they come from very not mysterious boxes that miraculously survives a ship exploding in space.

~ still a better story intro than Warcraft 3 Reforged ~

- Me, anno 6AM

Key Bindings

And Quick-Guide

In-Game Binds

W	**Ship Thrust Activate**
A	**Steer Left**
D	**Steer Right**
Space	**Shoot**
ESC	**Pause Game**

Score-Screen Binds

Any	**Exit Menu**

Quick Guide

These are the steps required to successfully start and operate the game

Before we Begin

This game is executed in undecorated full screen mode by default. It was developed with a 2560x1440 resolution with scaling in mind at an early phase. Regretfully some GUI elements ran into a VM-limitations during the last iterations of the development, causing the scaling to malfunction. Utilizing a lower resolution then mentioned above, the following graphical and functional elements might not behave as intended:

- Main menus transparent buttons on left hand side, incorrect location causing the buttons to appear disabled.
- Game & Main menu score screen, graphics shifted to incorrect positions.
- Game asteroids, ships, powerups, texts and mini map retains original size, causing the game-board to appear smaller then intended and limiting the players ability to respond to fast moving objects in time.

Playing the game

Clicking “New Game” in the menu initializes the game. The players ship is spawned in the middle of the screen, centered in the game world free of game objects. Using the key bindings on the left-hand side of this quick guide the player may traverse space and defeat enemy ships and asteroids alike. The goal of the game is to amass points and reach high scores while staying alive. Points are rewarded by destroying asteroids and enemy ships or by collecting the **Golden powerups** rarely dropped from enemy ships.

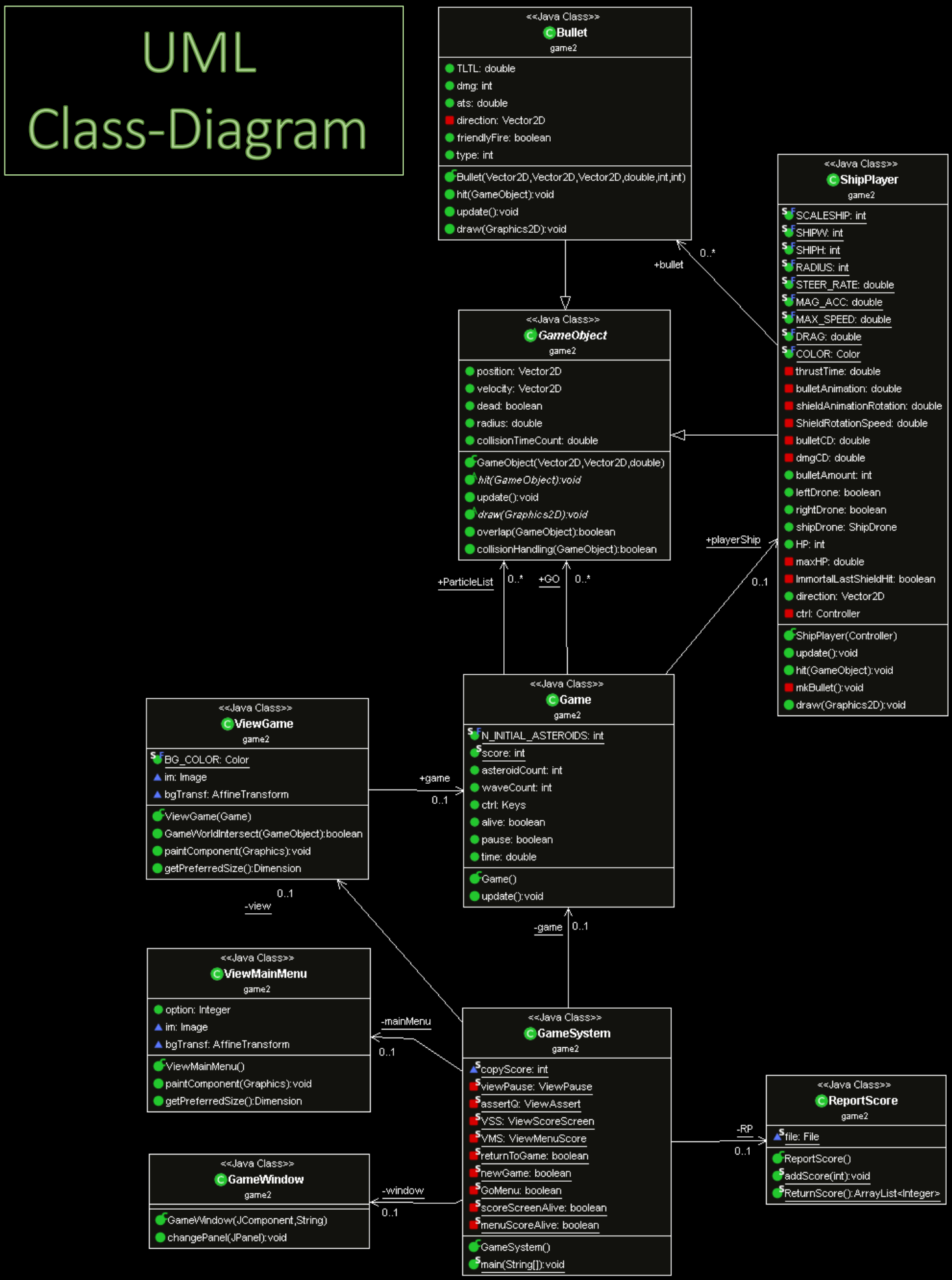
Points: **golden powerup > enemy ship > bigger asteroid > smaller asteroid**

The ship begins with a protective power shield that absorbs damage; this can be refilled with **Blue powerups**. **Green powerups** grants your ship a drone that fires at enemies while **Red powerups** increases the number of bullets your ship may fire.

Game Report

CE218-5-SP : Assignment

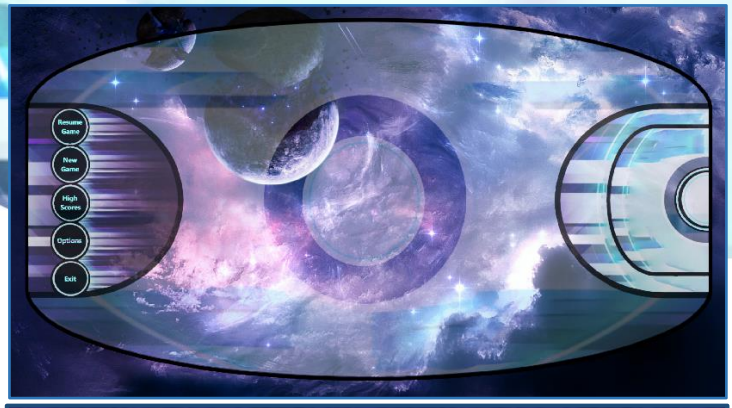
UML Class-Diagram



Game Breakdown

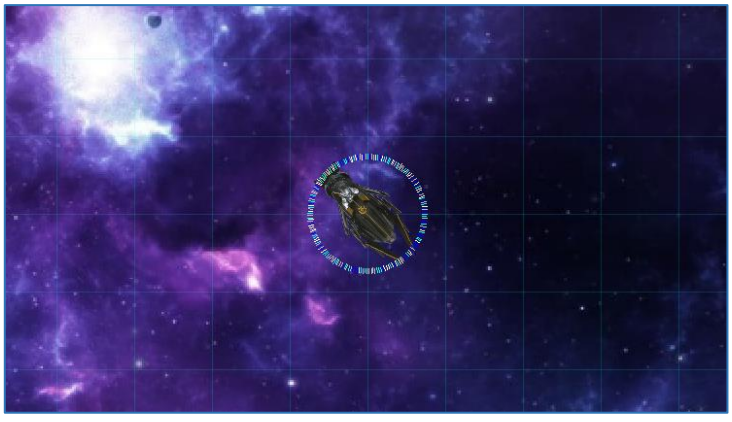
And Structure Analysis

ViewMenu



ViewGame

ShipPlayer



Class Diagram & Structural Analysis

Following is a description of the composition, purpose and structure of the different classes

Driver Main class: GameSystem.Java

Instantiates a GameWindow JFrame that acts as a container for the ViewMainMenu JPanel. Its main purpose is to decide what ContentPane to show in the GameWindow by using responses from the current JPanel.

If a new Game is started from ViewMainMenu, it instantiates a new game if a game is not currently running. Otherwise it instantiates an AssertQ JPanel to ascertain that the user wants to restart the current game.

This class acts as a driver for the game; It holds the game loop, all static JPanels, ReportScore and menu elements.

Camera JFrame: ViewGame.Java

Instantiated with a Game object. Responsible for painting all game related components and GUI within the bounds of its GameWorldIntersect() method. This decreases the number of paint() requests as it only paints elements in a certain distance to the players ship centred on the screen.

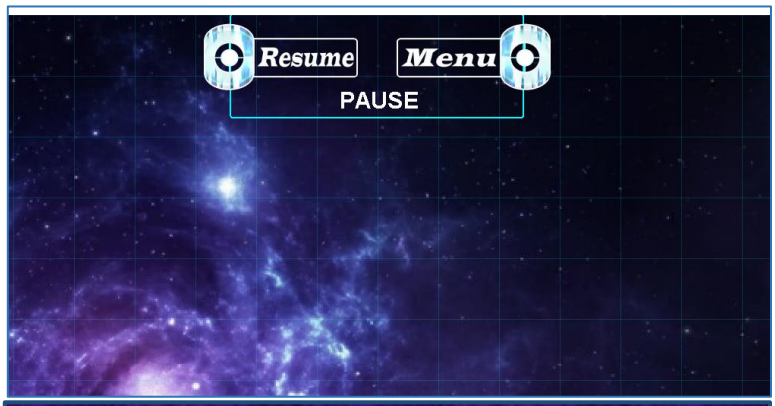
Game.Java

Updates GameObjects, removes dead flagged objects, sends collision check requests and plays collision particles if true. Checks ShipPlayer's, ShipDrone's & ShipEnemy's for instances of Bullet's and spawns these.

Spawns procedurally generated Asteroid's, ShipEnemy's & PowerUp's as time progresses. Outside of Asteroid children's spawned from death or time-left-to-live running out, most objects are spawned close to the border of the gameworld to simulate them "entering" your space.

ViewAssert

ViewPause



ViewScore - Menu

ViewScore – After Game

The JPanel ContentPanes

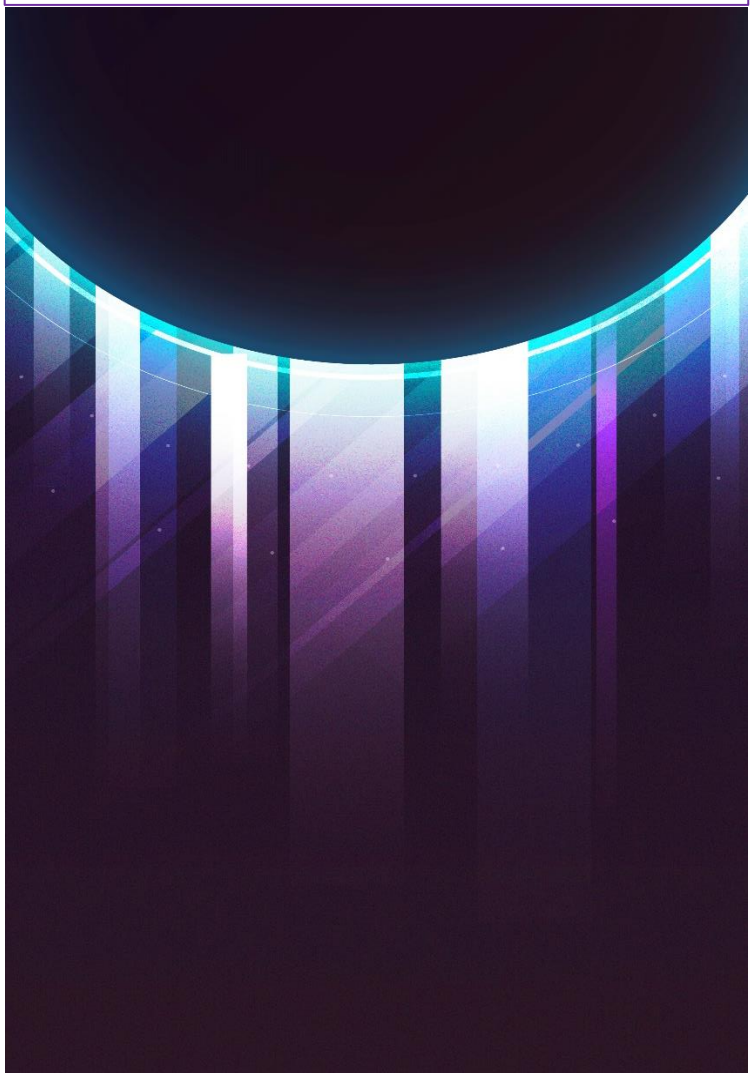
Including classes: ViewAssert, ViewPause, ViewMenuScore, ViewMainMenu, ViewGame & ViewScoreScreen.

These are the different JPanels acting as ContentPane's for the GameWindow.

ViewAssert shows a Questioner asking what the player want to do next in certain scenarios. If a game is running and an attempt to start a new game is made, the top pictures AssertQ is shown in the menus.

ViewPause pauses and resumes running games while also giving the player the option to return to the menu without destroying game progress. Do take note that no local saving method has been made, if the game is closed, the current progress is lost. On second thought there should have been another AssertQ if the player tries to exit the game with a game running, I forgot all about that.

ViewScoreScreen shows top 5 high score and current score if a game has just been completed.



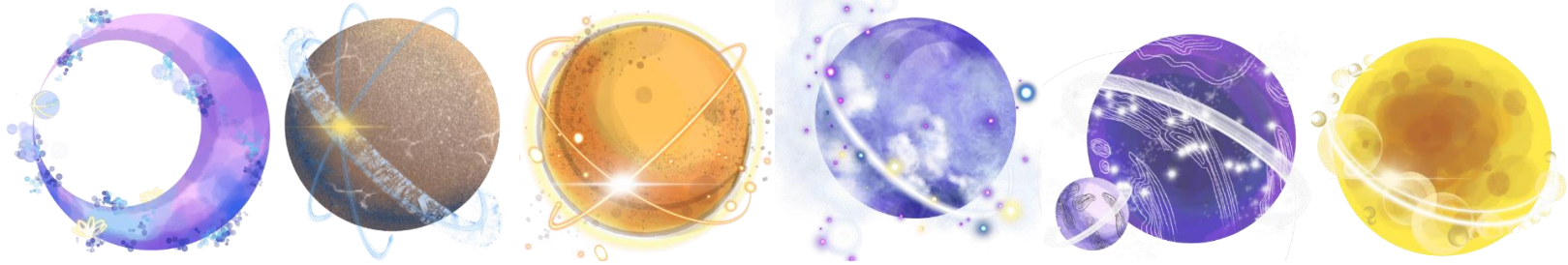
ShipPlayer

ShipEnemy

ShipDrone

Powerup

Asteroid



The GameObjects – Solid Objects

Including ShipPlayer, ShipEnemy, ShipDrone, Bullet, PowerUp, Asteroid

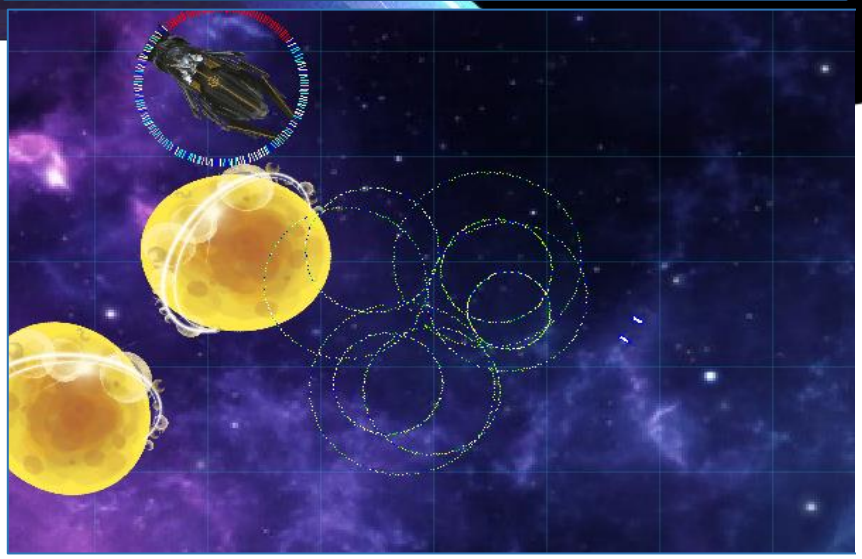
ShipPlayer is the user controlled GameObject. By far one of the biggest beasts when it comes to parameter tuning in this assignment. Uncountable hours later It ended up with an acceleration reaching top speed in 0.66 seconds, double bullets at spawn, an energy shield with the ability to absorb 30dmg and absorb one last hit covering any amount of damage before being destroyed, the ability to pick up and wield 2 ShipDrones simultaneously to outnumber your enemies. And let's not. I say not. Forget the awesome custom black and gold finish with a tint of dessert sunshine, brilliant.

ShipEnemy is your nemesis, it uses an instance of AiSeekAndDestroy to seek you out while avoiding Asteroids, get into firing positioning close to you and fire at you relentlessly. If by the off chance you or your drones manage to hit it a good few times (or if its radar breaks and it drives into too many of the big floaty rocks) it attempts to flee from the player. If you successfully hunt these enemies you have a good chance to pick up some absolutely LOVLY PowerUp's, you can't just leave them floating thought, as Enemy bullets can destroy them before you may pick them up.

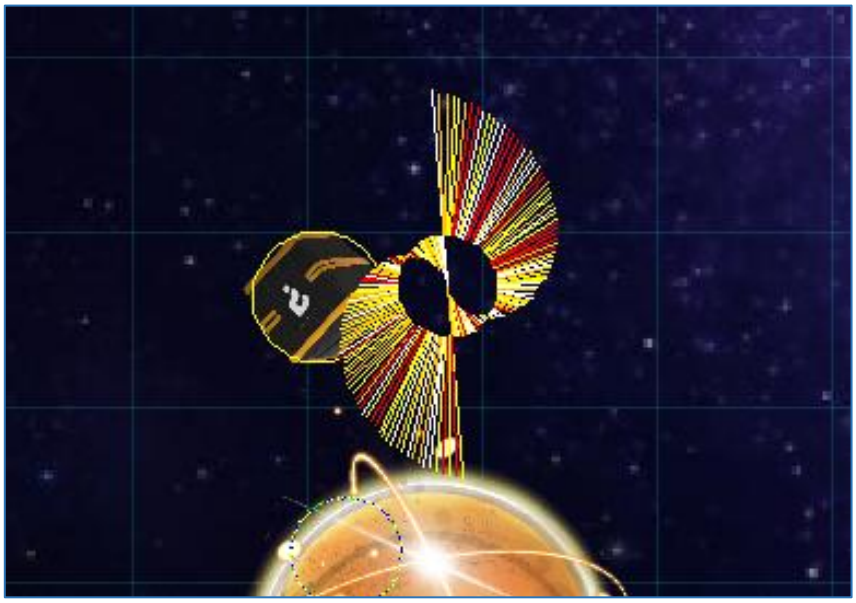
Asteroids, yes, the big floaty rocks hurling at you in the speed of light, commonly amassing more ShipEnemy kills then the User ever will. They move, collide, bounce, take damage, break on damage to spawn smaller asteroids. They even spawn after time to spawn an additional extra Asteroid as per the 2020 highlighted restriction.

Asteroids are the only graphics I haven't made myself. They were Created by "*588ku* - pngtree.com" under the terms of free usage for attribution https://pngtree.com/freepng/planetary-universe-nebula-free-collection_4627007.html

ParticleHit



ParticleRedSharingan



The GameObjects - Particles

Including ParticleHit & ParticleRedSharingan.

ParticleHit is used as the graphical element in collisions. Whenever two objects collide this graphic is shown as a couple of circle growing and shrinking rapidly while rotating. The circles are constructed from a myriad of lines in different colors, being drawn between an invisible inner circle and outer circle. This simulates a flickering circle and looks quite fancy. In a good effort I've done my best to avoid this being overly called due to multiple collision calls on the same GameObjects in quick succession with a cooldown code snippet. This in combination with a piece of code that despawns Asteroids on top of Asteroids before the game loop has a chance to update them twice takes cares of most of these excess calls.

ParticleRedSharingan. A particle effect that comes from manipulating chakra and innate bloodline abili….

- Enough of that. (Naruto reference)

Its created by painting 180 lines with shifting colors in a circular pattern with length being a relation between an invisible inner and outer circle, with the inner circle increasing in radius so it's bigger than the former outer circle before returning to its initial position. Whilst the entire thing rotates around its origin. Fancy.
Its used to mark the destruction of ShipEnemy.

Controller & Action classes

Including classes: Controller, ControllerAi, Action & ActionAi.

There's quite a bit of depreciated code here. User Controller and Action, parent classes to Keys, were separated from the Ai's version in an attempt to write more complex Ai's with the help of passing different objects to the controller constructors.

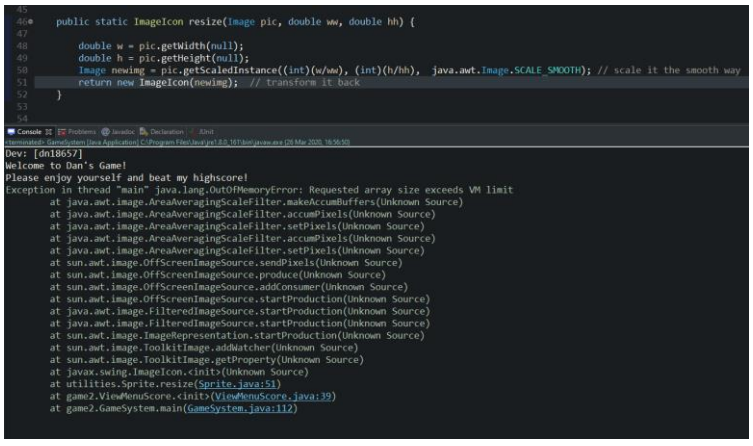
Ai & User Classes – Derived from Controller classes

Including classes: AiSeekAndDestroy, AiHelpPlayer & Keys

AiSeekAndDestroy was explained in full as the behavior of ShipEnemy.

AiHelpPlayer is the ShipDrones controller class. It searches the nearby vicinity for ShipEnemy Objects. If one is found withing 900 range the ShipDrone turns towards it, once in line of sight, 600 range, it begins to fire. Worth to notice is that this class does not handle the movement of the ShipDrone, as they are built to function as static turrets behind the ship, based on how many instances said ship may carry. The ShipPlayer GameObject may have 2 ShipDrones at a time.

Exceeding VM-limit



The Utility classes

Including classes: ImageManager, Sprite , SoundManager, Vector2D, ReportScore & Constants

ImageManager supports Sprite with static loading of pictures.

Sprite has the additional method of resize() which is used to scale pictures in what, used to be, an efficient way until I tried to rescale high resolution GUI elements, causing the game to be unable to rescale some GUI-elements properly on any resolution other than 2560x1440. The VM error seen in the picture below was the result at the last stage of testing, any potential fix or improvement to this and the resize method was dropped because of the time left to the deadline shrinking.

SoundManager is as is from the labs.

Vector2D has some additional methods in a mathematical attempt to sate the insatiable Vector2D JUnitTest "testAngle2". All for nothing none of the mathematical models to calculate this angle comes out correct for my attempts in Java. This amounts to no visual issues while doing angle calculation in the game and therefore I draw the conclusion that there is some underlying issue at work, but small enough to ignore for our development.

Depreciated classes

Including classes: Ship & JEasyFrame.

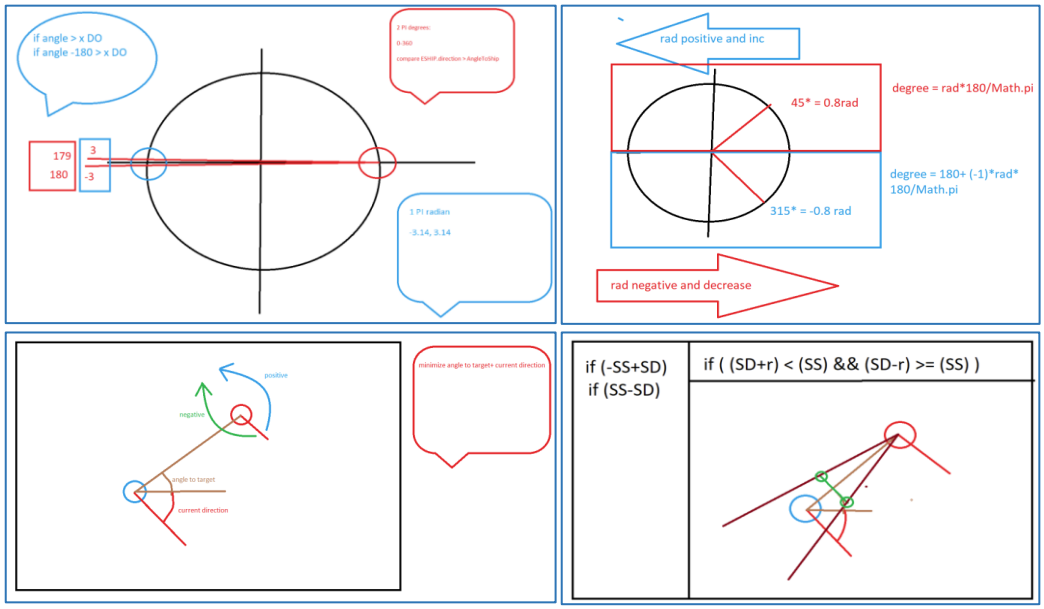
Early on in development I had plans to make an abstract ship class together with interfaces controller and action classes to define clear rules for how a ship may be generated. This was mainly because of the idea of using a drag and drop interface to add custom design to player ships and creating a ship factory in the menus/options so that the player could tweak their ship. For this to be realized I needed to have VM-safe picture scaling, this turned out to be an issue later in development and was dropped.

JEasyFrame was renamed and modified to GameWindow because of the need to switch content pane as my means of driving the applications window.

Parameter Tuning & Specificity

While tuning the myriad of parameters in the game project, there is a few that stands out as especially hard tuned.

The worst of the worst, Making the ShipEnemy's AI class AiSeekAndDestroy avoid Asteroids within a certain distance to it, with priority based on angle and distance. The main time consumer being the fact that our Vector2D class from labs are based on the (pi,-pi) range. It makes it almost unethical to tune based on math. Because a turnover from 179.9 degrees and 180.1 degrees varies the value from 3.14 to -3.14. regularly we would like the Ai to see if the ship is supposed to turn left or right and thrust or not to avoid objects. In this case we need to take the angle of the Ships direction into account, as this is the angle we would like to turn towards the target. The issue stems from Javas coordinate systems Origin being in the top-left corner of the JFrame. Working with (3.14, -3.14) in the 4th quadrant with an inverted Y-axis isn't optimal for angle calculations between points. If my understanding and extensive testing is correct and didn't miss any math our Vector2D angle finds the angle to the natural between 2 objects, to in addition take into account the direction of the ship we would need to work with some data on the form [angle between objects] + [angle ship is facing] (as seen in the lower-right paint picture). This isn't optimal, but good enough to duke out some decent moving Ai. I should have changed the (3.14,-3.14) range to a 360 degree or 2 pi system earlier to avoid these issues. Alternatively devised the math required to solve it with this range, but I couldn't tune this the way I wanted it.



Except for the Solid GameObjects tunings mentioned earlier the Particles were challengingly fun to tune and very rewarding once the correct math was in place. Especially simulating flickering circles with the help of straight lines between two different radiuses.

Tuning the Grid net shown between the background and the playable surface as well as its borders around the world space to move accordingly with the camera to simulate the ship moving instead of the Gameworld moving was a fun and rewarding challenge that really lifted the game to a new level graphically. I took inspiration from the labs mention of simulating stars with draw() and took it one step further.

Retuning almost all graphical variables with FRAME_HEIGHT, FRAME_WIDTH and WORLDSIZE in the hope of making a full screen game that could scale with screen resolution was an idea and priority that took place about halfway trough development. going back and forward between values countless times and still going through with this delivery empty handed left a sour taste behind. But as I've experienced earlier with rescaling websites and applications I am not entirely surprised and look forward to fixing the code bit by bit during my vacation time to make it scalable.

Self-Praise & Final words

So, we're finally here after 3 months of tinkering and duct taping.

I'm feeling vigorous having completed most of the things on my initial checklist for this game, even most of the things on the additional requirements I wanted have been implemented, revisited and improved.

All the graphical elements, including ships, asteroids and GUI has been switched over three times with different custom graphics. These were mostly made in Paint 3D with imported free-license material and a healthy amount of imagination and layering. The color theme of the GUI went from black-blue to green-white-blue to the final purple-black-white-blue.

While the current GUI feels a little bit detached from the game due to a lack of graphical connections or border elements and the Main menu buttons having no graphical response while being clicked due to implementing the button graphics on the same layer as the background, I believe it looks quite fancy, as this is, believe it or not, my first attempt at making any graphical object at all outside of 32x32 pixel graphic skins.

As I've mentioned already, the only thing I feel is greatly lacking would be the rescaling based on resolution backfiring in the later stages of development.

Minor issues would be:

- the angle() in Vector2D
- the ship not facing North at initialization
- no code to detect ahead of time if objects spawn on top of each other while being spawned in the border region. This is compensated by a cleanup code removing these object within a few iterations, 2 or 4. The only area being clear form this is the region the ShipPlayer spawns in at initialization as that was one of the mandatory requirements in the assignments.
- The Option item in the menu not being developed, its just a placeholder right now

My current thoughts and feelings now at the end of this project?

Fun, good learning curve, interesting, decent to good materials in labs, lacking info when it comes to information about changing content in an existing frame. Good amount of ideas for extra studies, flocking Ai was something I've started to read more into. I'm starting a Unity Cloud FPS dungeon crawler project in the summer with a few friends of mine, so this experience really hit it of as a good background check before diving into Unity's nitty and gritty libraries.

-I thank you for great teaching, great content and for taking your time to read this report to the end with the short amount of time allocated for grading.

~ Yours sincerely, Dan ~

Bonus Banana Nation Drone