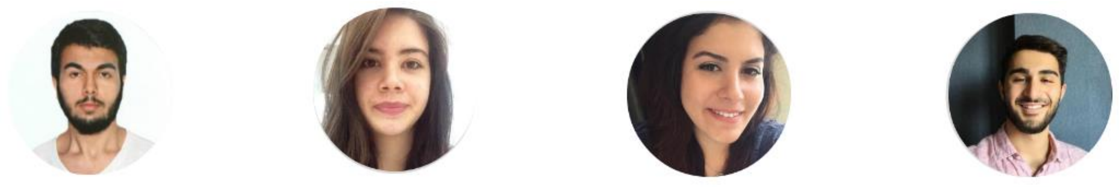
BILKENT UNIVERSITY

CS 319 – Object-Oriented Software Engineering

Design Report

Intergalactica

Group 18



Ahmet Burak Şahin 21301755

Melis Kızıldemir 21401184

Selin Erdem 21401244

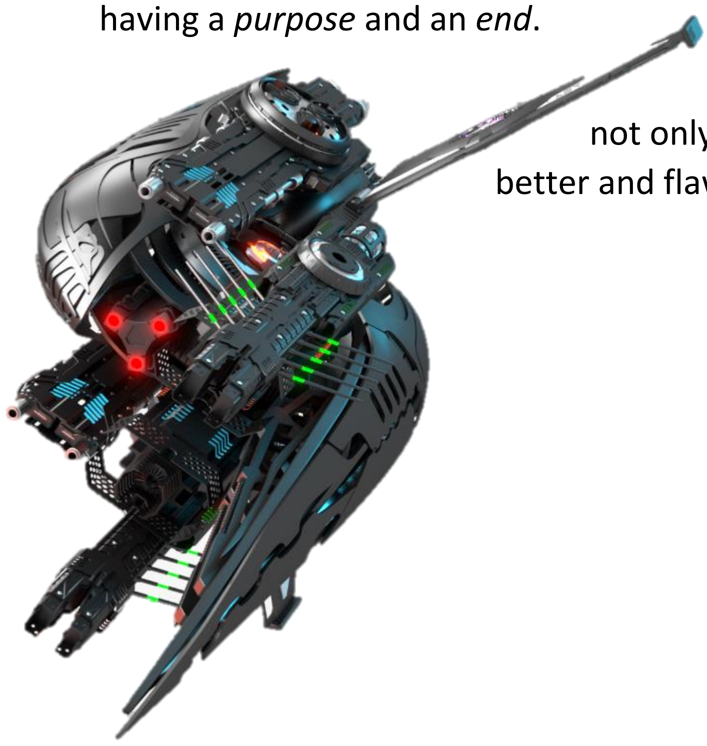
Ömer Sakarya 21301535

Table of Contents

1. Introduction

1.1 Purpose of the System

Among many versions of the *Galaga* [1], there is a missing element of ***adventure*** that might bring forgotten space-based games back to once-glorious days. *Intergalactica* is designed to fill that gap. It features some of the key concepts that was missing in the original implementation, such as travelling between planets, acquiring new ships throughout voyage and most importantly, having a *purpose* and an *end*.



Intergalactica project aims to furnish not only an adventurous, but also a graphically better and flawless experience to its aficionados.

Image by MillionthVector [2].

1.2 Design Goals

➔ ***Accessibility***

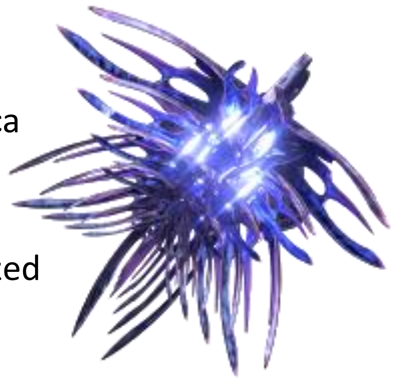
Intergalactica will be highly accessible since it is going to be free to use. We will distribute our game for the enthusiasts who may want to try.

➔ ***Efficiency:***

The main design goal in Intergalactica is being able to provide a smooth gaming experience to the user along with a dazzling UI design.

➔ ***Maintainability:***

Almost in every step of software development, a major care must be taken to ensure the final product is maintainable. So does the development of Intergalactica have taken maintainability as pivotal factor. Entire project is being kept under the VCS (version control system –Git) to ensure that the project remains organized and in any case of failure in the system, issue can be addressed by tracing the development background.



➔ ***Portability:***

Intergalactica is being implemented in Java platform. This specific language is chosen because it runs regardless of the underlying system as long as the machine contains Java Virtual Machine (JVM).

➔ ***Reliability:***

For any run-time issues, any kind of exception will be handled thoroughly on the fly in order not to interrupt the gaming experience. Severe crashes in general, might as well corrupt the data that was collected previously by the user, however, Intergalactica's one of the core design goals is to be reliable and robust to not to result in data loss or cause any damage to the system.

➔ ***User-Friendliness:***

Intergalactica will have very plain, easy to use and intuitive interface even that –although we will include a brief one–a tutorial wouldn't be necessary for any regular user to get the hang of it.

All Images in this page by MillionthVector [2].
Images are 3D imaginary depictions of some of our alien ships.

2. Software Architecture

2.1 Subsystem Decomposition

The decomposition of our subsystems is depicted as follows:

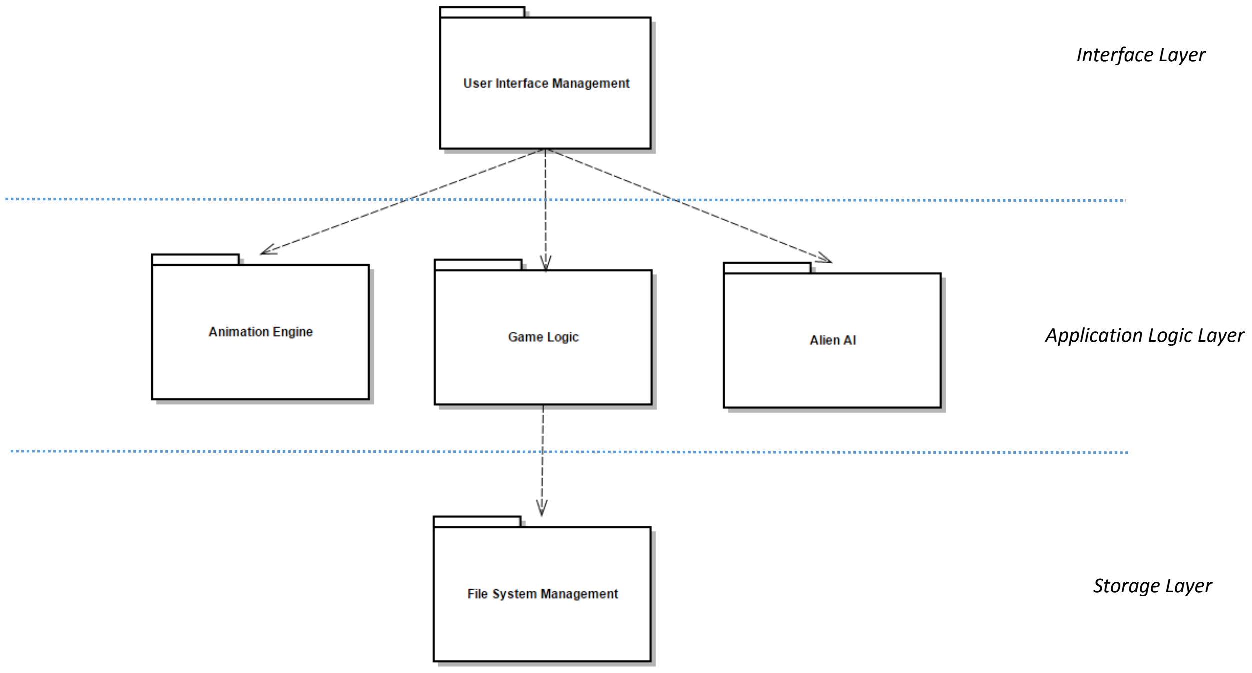


Diagram above shows Subsystem Decomposition of Intergalactica.

Intergalactica's subsystems suit well with the three-tier architectural style. Every boundary and user-related operations are included in the Interface layer (top) that deals with the user interface and input issuing to the lower layer related tasks. It is the closest layer to the end-user. Below that, we have Animation Engine, Game Logic and Alien AI that makes up the Application Logic Layer (middle), which is responsible for controlling and managing of the main events during the execution. Finally, at the bottom, there lies the Storage Layer, which, in our case, provides basic functionality to save and restore high scores that was accomplished previously by the users. Intergalactica doesn't highly make use of this layer, simply because every game session won't be restored after exiting the program due to the core design of the game. All packages' responsibilities and services will be discussed in Section 3.

2.2 Hardware/Software Mapping

Our system shall be realized via both software and hardware point of view. From software perspective, since it is going to be developed in Java platform, any computer having Java Runtime Environment (JRE) will be able to run Intergalactica, independent of the underlying OS.

From the hardware point of view, the requirements are relatively minimal. I/O devices necessary to play the game consists of: a PC, a mouse, a keyboard and a monitor. Regardless of the size of the monitor, Intergalactica will run correctly, because the software will be window-based.

2.3 Persistent Data Management

Nonvolatile data management will be handled in File System Management subsystem. Using the User's hard drive, only top 10 high scores along with the scorers' names will be stored. The file format we have chosen for the ease of parsing is .csv (comma separated value). Moreover, since each entity will have its own appearance through a picture, each .png file will also be permanently stored. Finally, for background music, we will store .waw files in the hard disk.

2.4 Access Control and Security

In Intergalactica there's no authentication process and users will not, by any means, be prompted to share any particular personal information.

2.5 Boundary Conditions

Initialization:

- Intergalactica is downloaded from the source and placed somewhere in the hard disk drive.
- User starts the program.
- System boots up and User starts playing the game.

Termination:

- In any page, User presses ESC key which makes a menu pop up.
- From the Pop-up menu, User selects Quit.

Failure:

- Since Intergalactica does not support save/load feature for any game, in case of a failure, all the progress achieved by the User will be lost.
- System also can face technical faults during processing alien motion or animation generation.
- In the case of failure, system will halt itself and display information through a pop-up window once it guarantees that persistent data is safe.

3. Subsystem Services

➔ *Interface Layer*

User Interface Management:

This subsystem is responsible for controlling the Graphical User Interface. To elaborate, it directs inputs to the corresponding subsystem and in charge of reflecting the response generated by either Game Logic, Alien AI or Animation Engine subsystems upon any stimulus entered by the User.

➔ *Application Logic Layer*

Animation Engine:

Since Intergalactica requires heavy use of animation, a separate engine needs to be abstracted out from the game logic in order to see the whole system clearer. This subsystem is in charge of every moving object in the game and providing necessary updates, calculations (excluding aliens). Whenever, a spaceship moves, gets destroyed or an item drops, graphical information (which sprite) and necessary information (where It was, how should it go) is passed down to the user interface via this subsystem.

Alien AI:

Evidently, user is supposed to fight against semi-smart opponents which is crucial for the game to be entertaining. This subsystem provides information to Game Logic and User Interface Management subsystems so that each alien can make its own choice of action. Those choices include: *which path to take, how to dodge, when to shoot.*

The path taken by aliens is calculated by linear and quadratic Beziér curves with following formulas:

$$\boldsymbol{B}(t) = (1-t)\boldsymbol{P}_0 + \boldsymbol{tP}_1, \quad 0 \le t \le 1$$

$$\boldsymbol{B}(t) = (1-t)^2\boldsymbol{P}_0 + 2(1-t)t\boldsymbol{P}_1 + t^2\boldsymbol{P}_2, \;\; 0 \le t \le 1$$

Of course this calculation will not be dynamic, pre-defined formulation is necessary on account of potential heavy workload which might slow the whole execution down. There will be several formulations (hard-coded polynomials) and aliens will choose one that suits the best. In similar fashion, upon a decision of dodge, this subsystem will choose most appropriate path to take in order to dodge that user-shot, expectedly if there's no way to dodge it, shot will be taken indispensably.

For shooting, Alien AI will calculate the position and path currently being taken by the both the particular alien and the user-ship, and will try to shoot down the user's ship. Path of a shot will always be linear and downwards starting from where it was initiated.

Game Logic:

This subsystem is the core of our game, every management is centralized around this component. It is responsible for maintaining the overall game, responding other subsystems about which action should be taken, and carries the game information.

➔ *Storage Layer*

File System Management:

At the beginning of each execution, previous game's high scores get restored for the purposes of displaying them when user desires. Furthermore, at the end of each game user is prompted to enter his/her name and thus the score he/she made gets saved via this management system. Game Logic is the only subsystem that interacts with this component.

4. References

[1] “Galaga.” Wikipedia. Wikimedia Foundation, n.d. Web. 11 Oct. 2016. (https://en.wikipedia.org/wiki/Galaga)

[2] “Free Sprites.” MillionthVector:. N.p., n.d. Web. 11 Oct. 2016. (http://millionthvector.blogspot.com.tr/p/free-sprites.html)