BİLKENT UNIVERSITY

CS319 – Object Oriented Software

FALL 2015

“RUN FROM SHAPES”

ANALYSIS AND DESIGN REPORT

NOVEMBER 20,2015

GROUP MEMBERS

Ebubekir Karabuğa

Elif İdiz ARAT

Caner Akça

Çağlar Alkış

SECTION-GROUP NO: 3-1

Table of Contents

[1. Introduction 3](#_Toc435743645)

[2. Requirement Analysis 3](#_Toc435743646)

[2.1. Game Overview 3](#_Toc435743647)

[2.1.1 List of Bonusses 4](#_Toc435743648)

[2.2. Functional Requirements 4](#_Toc435743649)

[2.3. Nonfunctional Requirements 4](#_Toc435743650)

[2.3.1. User Friendly Interface 4](#_Toc435743651)

[2.3.2. Variety 5](#_Toc435743652)

[2.3.3. Game Speed 5](#_Toc435743653)

[2.3.4. Animations 5](#_Toc435743654)

[2.4. Constraints 5](#_Toc435743655)

[2.5. Scenarios 6](#_Toc435743656)

[2.5.1. Use Case: Play Game 6](#_Toc435743657)

[2.5.2. Use Case: Get Help 8](#_Toc435743658)

[2.5.3. Use Case: Change Difficulty 9](#_Toc435743659)

[2.5.4. Use Case: Change Color 10](#_Toc435743660)

[2.5.5. Use Case: Exit 11](#_Toc435743661)

[2.6. Use Case Model 12](#_Toc435743662)

[2.7. User Interface 13](#_Toc435743663)

[2.7.1. Navigational Paths 13](#_Toc435743664)

[2.7.2. Screen Mock-Ups 14](#_Toc435743665)

[3. Analysis 19](#_Toc435743666)

[3.1. Object Model 19](#_Toc435743667)

[3.1.1. Domain Lexicon 19](#_Toc435743668)

[3.1.2. Class Diagram 21](#_Toc435743669)

[3.2. Dynamic Models 22](#_Toc435743670)

[3.2.1. Statechart Diagram 22](#_Toc435743671)

[3.2.2. Sequence Diagrams 26](#_Toc435743672)

[3.2.3. Activity Diagram 31](#_Toc435743673)

[4.Design 32](#_Toc435743674)

[4.1 Design Goals 32](#_Toc435743675)

[4.2 Sub-System Decomposition 34](#_Toc435743676)

[4.2.1 Detailed Sub-System Decomposition 36](#_Toc435743677)

[4.3 Architectural Patterns 37](#_Toc435743678)

[4.4 Hardware/Software Mapping 37](#_Toc435743679)

[4.5 Addressing Key Concerns 38](#_Toc435743680)

[4.5.1 Persistent Data Management 38](#_Toc435743681)

[4.5.2 Access Control Security 38](#_Toc435743682)

[4.5.3 Global Software Control 38](#_Toc435743683)

[4.5.4 Boundary Conditions 39](#_Toc435743684)

[5. Conclusion 40](#_Toc435743685)

[6. Reference 41](#_Toc435743686)

# Introduction

We decided to design and implement a game called *Run from Shapes*. In our game, aim of the player is escape from the falling down shapes. The game will be a desktop application and will be controled by mouse. This report contains an overview of the game, describes the basic gameplay elements and rules of the game. Then it describes functional requirements, non-functional requirements, use-case’s models including scenarios and use-case diagrams.

# 2. Requirement Analysis

## 2.1. Game Overview

Purpose of this game is stay alive while shapes are falling down from the top of the platform. In our game, our charachter icon is a circle and it will be controlled by mouse move.There will be restricted platform that during the game, it can not be extended or shrinked.Shapes will fall down from the top of the platform and they can pass beyond the platform but user must remain in the boundaries of platform. Shapes will be created randomly which means that shapes can be any polygons and while those shapes falling down from the top of the platform, they will be spin around theirselves which makes harder to play the game for the player. Score of the player will depend on survival time of the player. Score will be increase linearly while player keep avoid from falling down shapes.Whenever score of the player reachs multiples of hundred(ie. 100,200,300,..) level of the player will increase by one. Maximum level of the game will be five and after a player level five, speed of the game remain constant regadless of the player score. Also there will be special bonuses which is represented by special shape which is a star, and if player can collect these stars, player gain bonus abilities or features which makes game easier for the player. If player crush any shape that fall down from above(except shape of star), player lose one life point and if player has used all of his lives, than game will be over and the player can see his/her total score in the pop-up screen. There will be multiple color option for charachter icon and player can easily choose his/her charachter icon color. Also for player who prefer challenge, there will be a difficulty option in our game. There will be three difficulty settings in our game which are easy medium and hard. Whenever player choose one of those options , his/her icon can be shrinked or extended, its depends on difficulty choise of the player. In addition to those features, there will be help button in our game which helps player to understand to game play, level up system and bonuses.

### 2.1.1 List of Bonusses

* **Additional life:** It increases the lives of the player by one.
* **Smaller icon:** Whenever player collect this bonus, his/her charachter icon gets shrinked.
* **Bonus score:** Players score increase by 10.

**Important Note\*\*:** When player hits any object that falling down from the top of the platform, all of the bonuses will be canceled.

## 2.2. Functional Requirements

* User will be able to control the character icon by mouse move.
* User can change the settings of the game. Available settings are ;

**1.Difficulty:** This option allows the player to choose size of the circle that represent the player.

**2.Color:** By clicking this button, user can select the order of the circle.

* Our game has a score system which is based on time that player survive, Score of the player lineary increase while player avoid from falling down polygons.
* Game will offer level system which is depend on player score. If player has achieve multiples of hundred points(100,200,300,400,500) level of the game will increase by one which means that object will fall down faster than before. 500 points is equals to level five which is the maximum level of the game and after that game speed remain constant regadless score of the player.

## 2.3. Nonfunctional Requirements

### 2.3.1. User Friendly Interface

Game will be consist of user-friendly interface, user can easily interact with the game and its easy to play the game. As a group, we care player desires which is feel comfortable while playing the game and for achieving that we developed understandable interface to players.

### 2.3.2. Variety

Since we developing a game that based on shapes, we are planning to use some smooth dynamic animations in order to show objects movement clearly.

### 2.3.3. Game Speed

Game speed changes from 10 fps to 60 fps respectively.

### 2.3.4. Animations

Vertex coordinate of the shapes is calculated according to mathematichal(sin and cosine functions) calculations in order to achieve smooth rotations of shapes.

## 2.4. Constraints

The game will be implemented in Java. We used few libraries from Java for some classes and methods. For example, in Shape class for updatingCoordinate method we benefited from “java.lang.Math”. Also, we use “javax.swing” library for processing graphics. Also, we use “java.awt” and “java.awt.event” for mouse listener. For time calculations, we used “java.util.Timer” and “java.util.TimerTask” .Resolution is 1000x800 in game which gives a huge space for player.

## 2.5. Scenarios

### 2.5.1. Use Case: Play Game

|  |  |
| --- | --- |
| **Use Case Name** | Play Game |
| **Primary Actor** | Player |
| **StakeHolders and Interests** | None |
| **Pre-Condition** | Game should be activated. |
| **Post-Condition** | After players has lost all of his/her lives in any level, his/her score will be shown by the system in a pop-up screen. |
| **Entry Condition** | Player selects “Play Game” button from Main Menu. |
| **Exit Condition** | Player has lost all of his/her lives in any level. |
| **Flow Of Events** | 1. Game is started after the ***“Play*** ***Game”*** button is clicked. 2. Player starts from the first level which is minimum speed of the gam and player score is 0 in the begginning. 3. Shapes will start to fall down from the above of the platform. 4. After shapes start to fall down from above of the platform, player will start to move the mouse in order to escape from the shapes that are coming through the charachter icon of the player. 5. Score of the player will increase automaticly and linearly while he/she avoid from crush any falling down polygons. 6. Shape which is created newly will enter the platform zone. 7. If player succesfully escape from shapes and if player’s score reach next levels threshold point then level of the game will increase by one.. 8. Player will move next level. 9. Player plays until he/she consumes all of lives of the circle. |

### 2.5.2. Use Case: Get Help

|  |  |
| --- | --- |
| **Use Case Name** | Get Help |
| **Primary Actor** | Player |
| **StakeHolders and Interests** | Player aims to get help from the system in order to learn how to play game. |
| **Pre-Condition** | Player should be in main menu. |
| **Post-Condition** | None |
| **Entry Condition** | Player selects “Help” button from main menu |
| **Exit Condition** | Player should press the “OK” button from the pop-up screen. |
| **Flow Of Events** | 1. Player selcets “Help” from the main menu. 2. The system displays the help documentation about the game. 3. After reading the documentation, player returns to the main menu. |

### 2.5.3. Use Case: Change Difficulty

|  |  |
| --- | --- |
| **Use Case Name** | Change Difficulty |
| **Primary Actor** | Player |
| **StakeHolders and Interests** | Player desires to change the game difficulties. System updates the settings which are changed by the player. |
| **Pre-Condition** | For first running, difficulty will be set as default. Player should be in the main menu in order to change game difficulty. If Player changes difficulty, adjusted settings will be saved and used by system. |
| **Post-Condition** | Difficulty of the game updated. |
| **Entry Condition** | Player selects “Difficulty” button from menu. |
| **Exit Condition** | None |
| **Flow Of Events** | 1. Player choses to difficulty from main menu. 2. The system displays different difficulties on the gameplay. 3. Player configures the settings according to his/her comfort. 4. The system saves corresponding changes in order to player play with desired difficulty. 5. Player returns to main menu. |

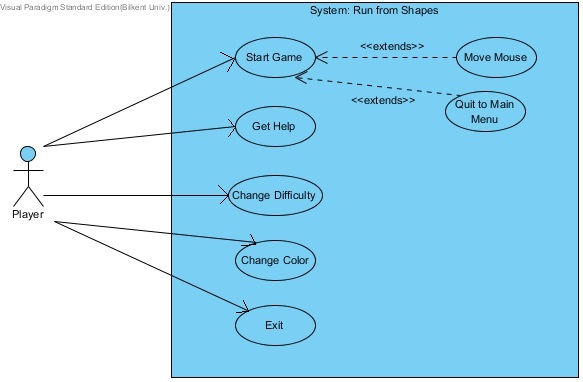
### 2.5.4. Use Case: Change Color

|  |  |
| --- | --- |
| **Use Case Name** | Change Color |
| **Primary Actor** | Player |
| **StakeHolders and Interests** | Player desires to change in game color of his/her charachter icon.System updates the settings which are changed by the player. |
| **Pre-Condition** | User must be in main menu. |
| **Post-Condition** | Color of the his/her shape updated. |
| **Entry Condition** | Player should select desired color from main menu by click the left button of the mouse. |
| **Exit Condition** | None |
| **Flow Of Events** | 1. Player choses to desired color by using color button. 2. The system saves corresponding changes in order to player play with desired color. |

### 2.5.5. Use Case: Exit

|  |  |
| --- | --- |
| **Use Case Name** | Exit |
| **Primary Actor** | Player |
| **StakeHolders and Interests** | Player desires to exit from game. |
| **Pre-Condition** | None |
| **Post-Condition** | Game will be closed by the system. |
| **Entry Condition** | Player selects “Exit” button from main menu or player clicks default close operation which is located in the northeast corner of the game . |
| **Exit Condition** | None |
| **Flow Of Events** | 1. Player selects “Exit” button from main menu. 2. System will close the game. |
| **Alternative Flow Of Event** | 1. If player request exit from the game while keeping play the game,    1. Player clicks the default close operation which is located in the northeast corner of the game .    2. System will close the game. |

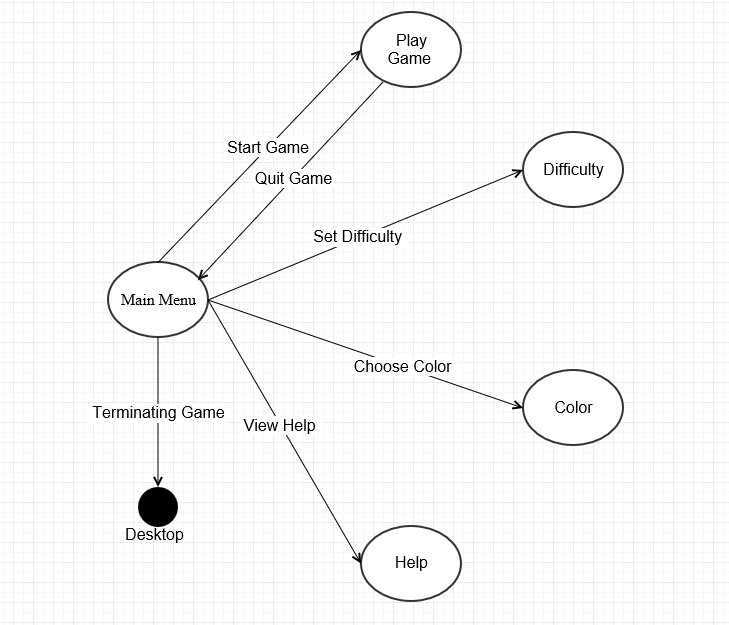
## 2.6. Use Case Model



**Figure 2.6- Shows use case model of the Run from shapes)[1]**

## 2.7. User Interface

### 2.7.1. Navigational Paths

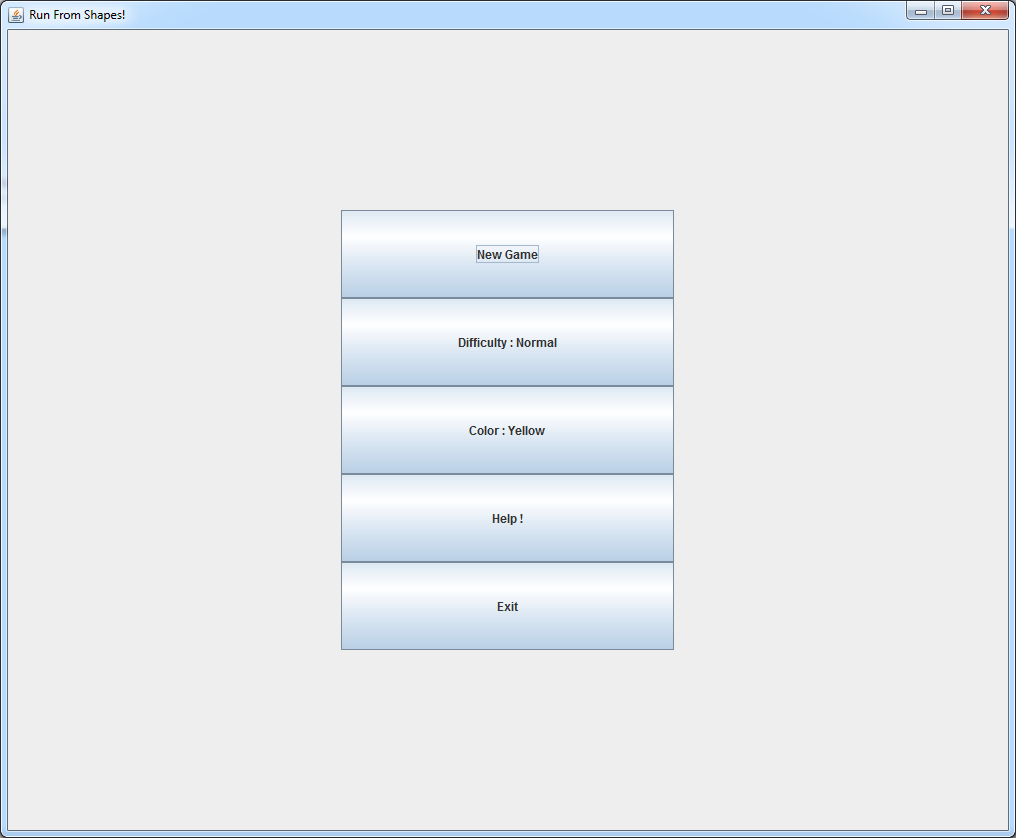


**(Figure 2.7.1 Shows Navigational Paths)[1]**

### 2.7.2. Screen Mock-Ups

#### 2.7.2.1. Main Menu

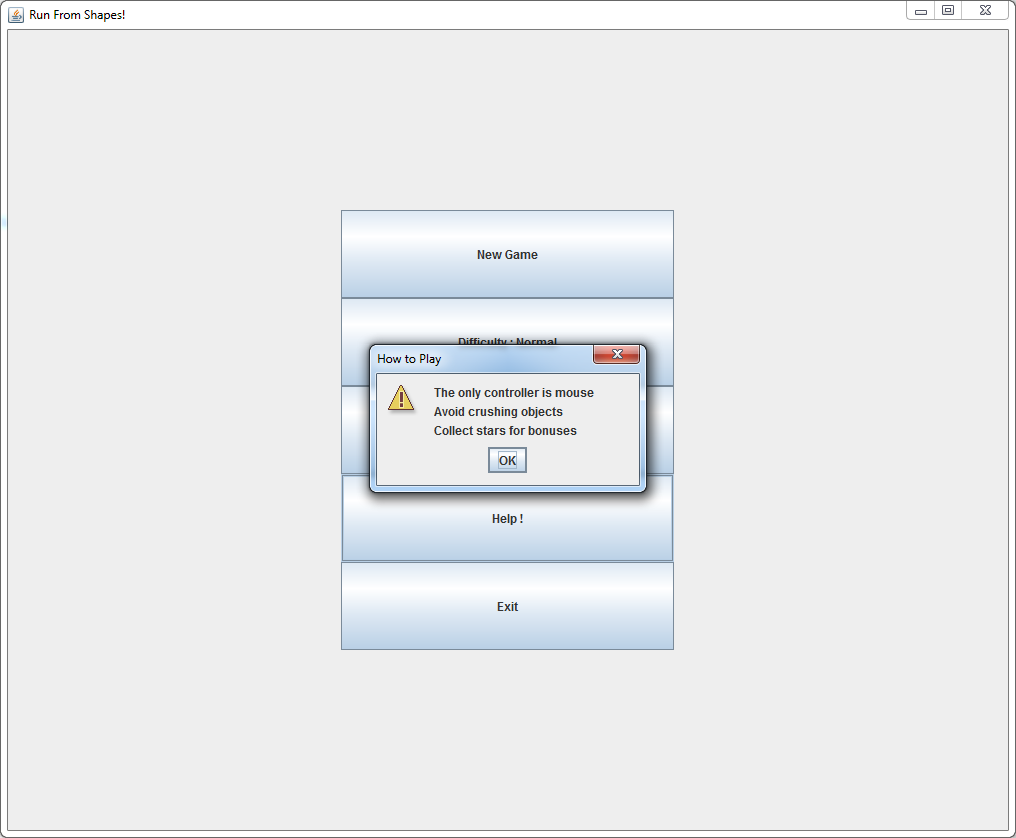
When application begins to run, Player will see main menu screen. Main menu screen shows five options to Player which are New Game, Difficulty, Color, Help, and Exit . (Figure 2.7.2.1)



**(Figure 2.7.2.1 – Shows Main Menu Interface)**

#### 2.7.2.2. getHelp

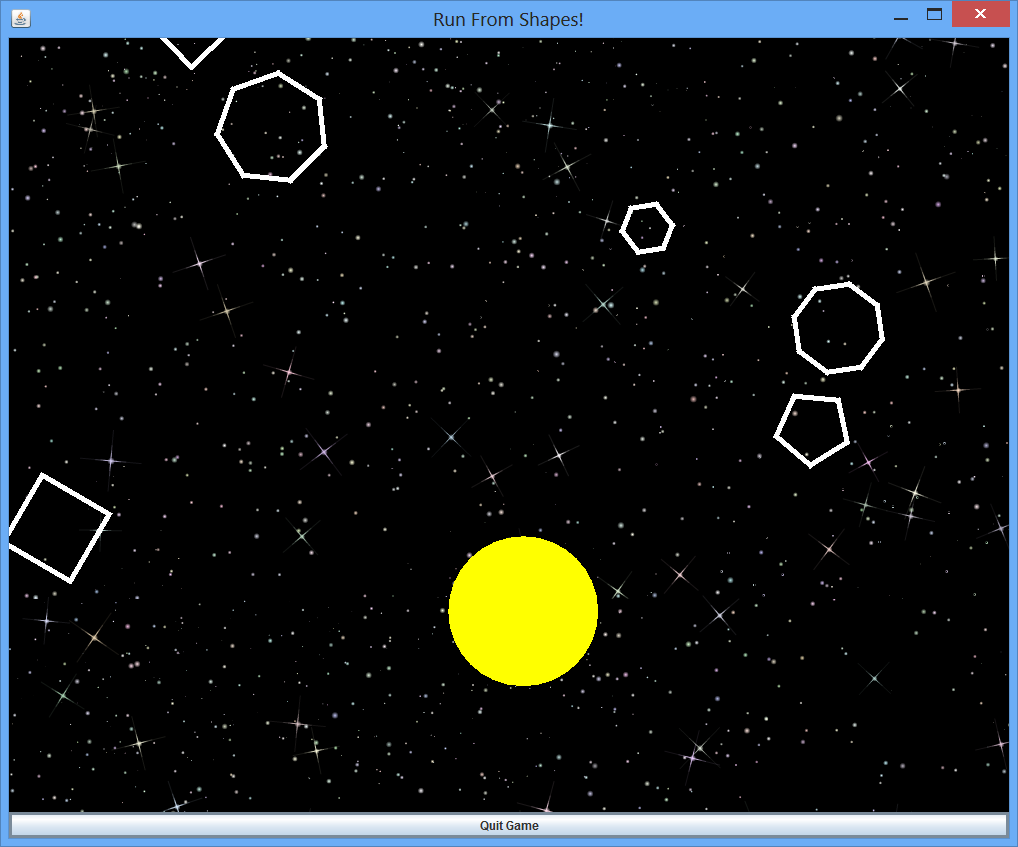
When player clicks help from main menu, new screen will appear and in this screen, player can learn how the play this game. After player finish reading the gameplay than he clicks the “OK” button from the pop-up screen in order to return to main menu.(Figure 2.7.2.2)



**(Figure 2.7.2.2-Shows Help Screen )**

#### 2.7.2.3. Gameplay

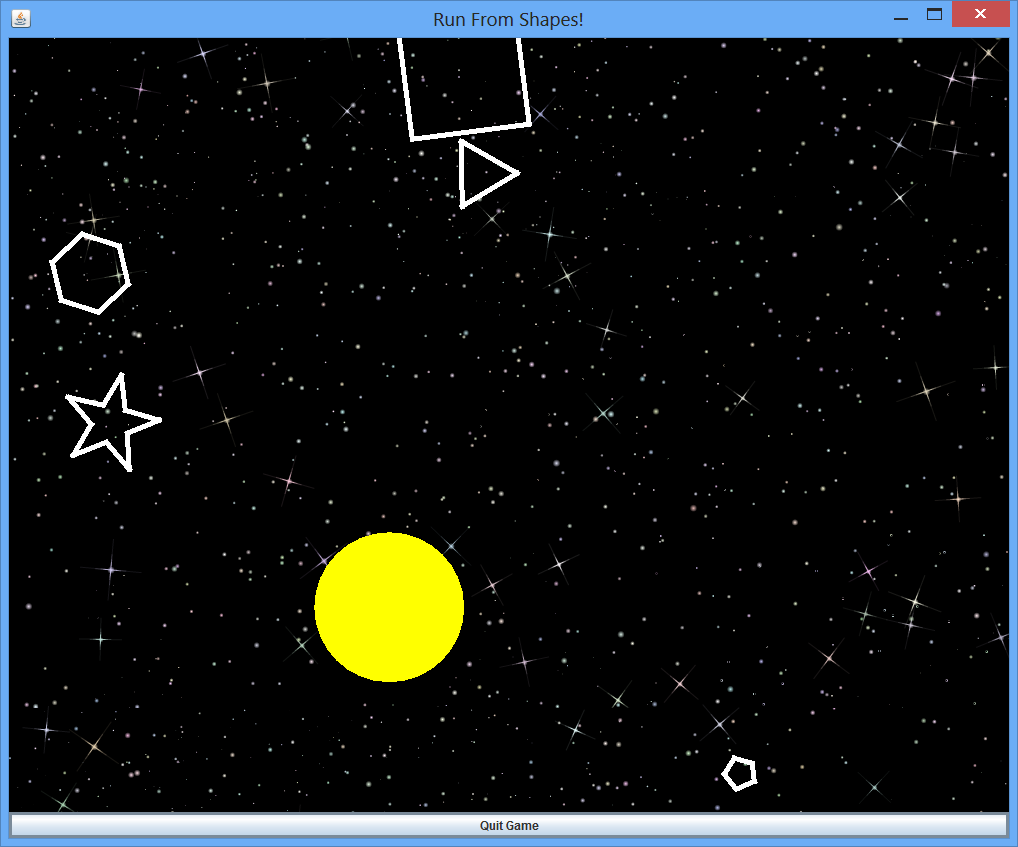
After player clicks new game option from the main menu, game starts to play and shapes will be automatically created and fall down from the top of the platform. In figure 2.7.2.3, yellow circle represents players charachter icon and he/she tries to avoid any collision from shapes above him by using mouse move.

****

**Figure(2.7.2.3- Shows game playing main screen )**

#### 2.7.2.4. Bonusses

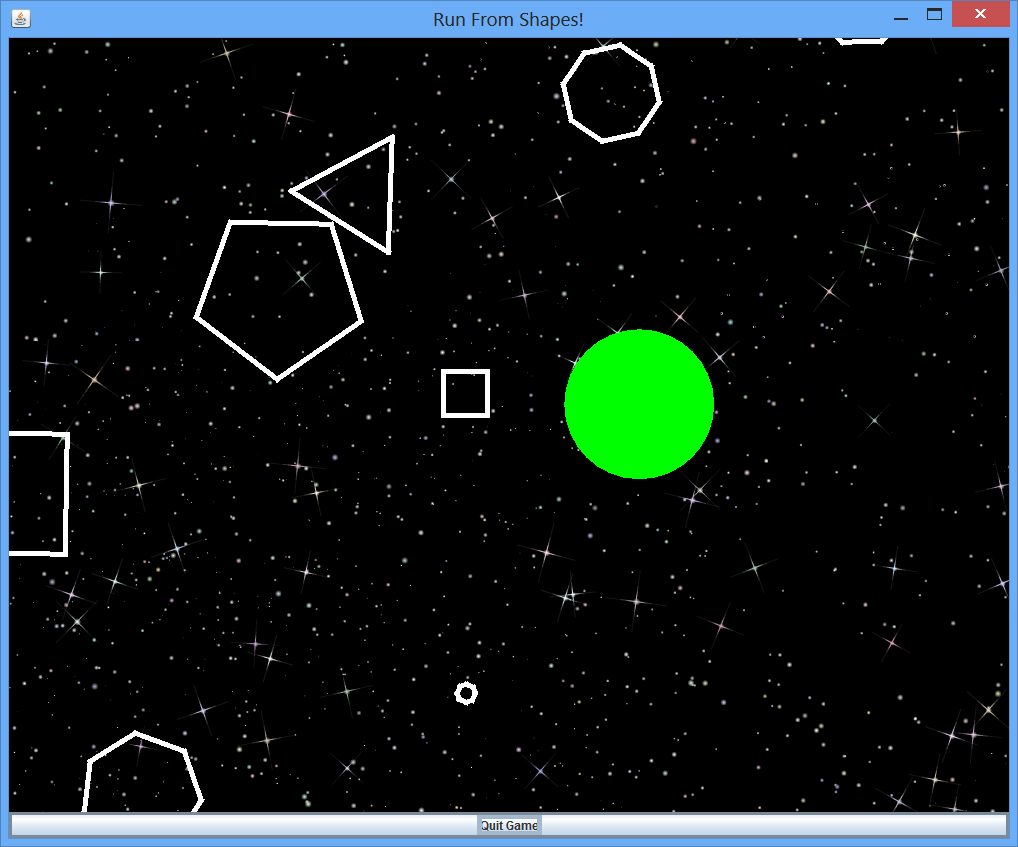
While player keep playing the game a specific shape which is a star falling down from the top of the platform randomly. Since star is also a shape, it created randomly and its creation has same probability with other shapes. Whenever player collects these stars, he/she gains some bonuses and these bonuses might be extra life,extra score or smaller charachter icon. Star is shown in Figure 2.7.2.4



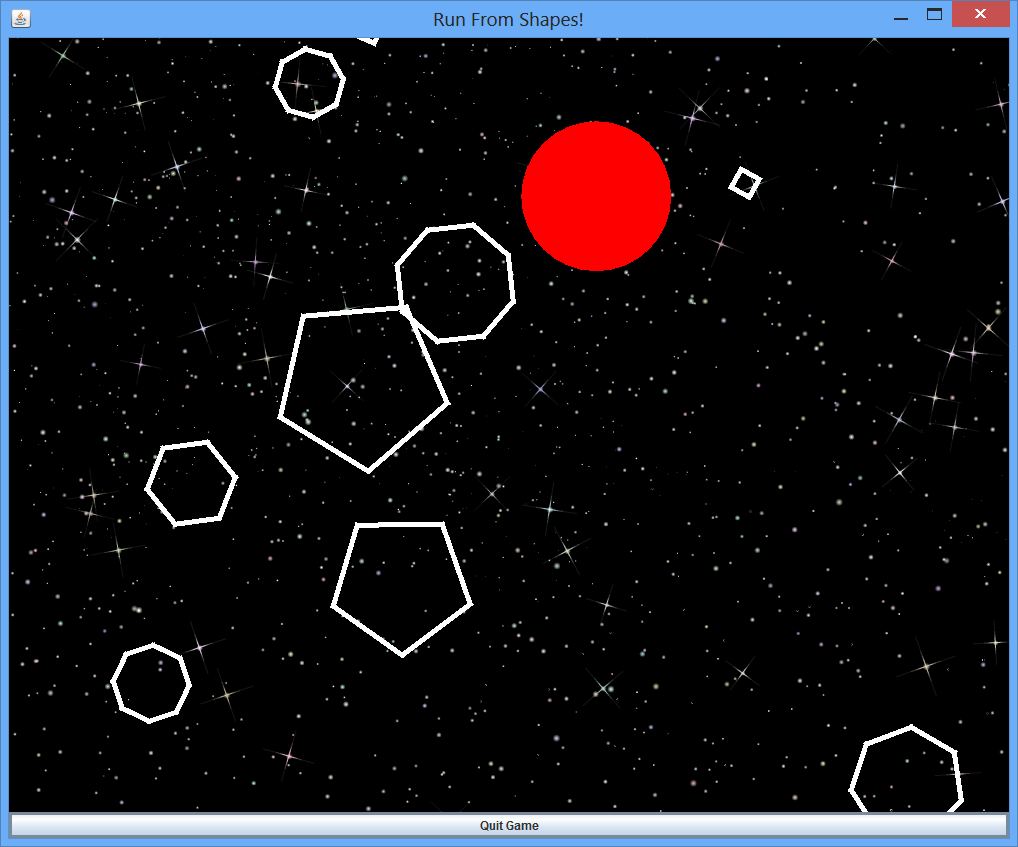
**Figure(2.7.2.4-Shows star bonus)**

#### 2.7.2.5 ChangeColor

Before starting game, if player wants to change color of the main character which is circle in our game, with just clicking Color option from main menu.We have limited color options that are red, green and yellow.



**(Figure 2.7.2.5/1 –Shows green circle option while player keeps playing)**



**(Figure 2.7.2.5/2 –Shows red circle option while player keeps playing)**

# 3. Analysis

## 3.1. Object Model

### 3.1.1. Domain Lexicon

During the design of Run from Shapes more than one comprehensive domain had to be examined : game engine, user interface,content and logic.

**a)Game Engine**

This part of the system which does not interact with the user.Game engine is about interactive control objects and ultimately changes in other objects, it is completely seperated from game logic and game contents.

* **Collusion :** Collision refers interaction between player’s circle and the other shapes that falls down.
* **Rotation :** A computer graphics term, rotation specifically refers to thev aciton of the rotating a shape or image according to specific origin.
* **X Width :** A graphical term defines the x coordinate of a drawable object in space.
* **Y Length :** A graphical term defines the y coordinate of a drawable object in space.
* **Update :** This refers to applying all changes in the game logic to the game objects which are shapes in the game.

**b)Game Control**

This part of the system deals with user interaction and its effects on the both game engine and game logic.

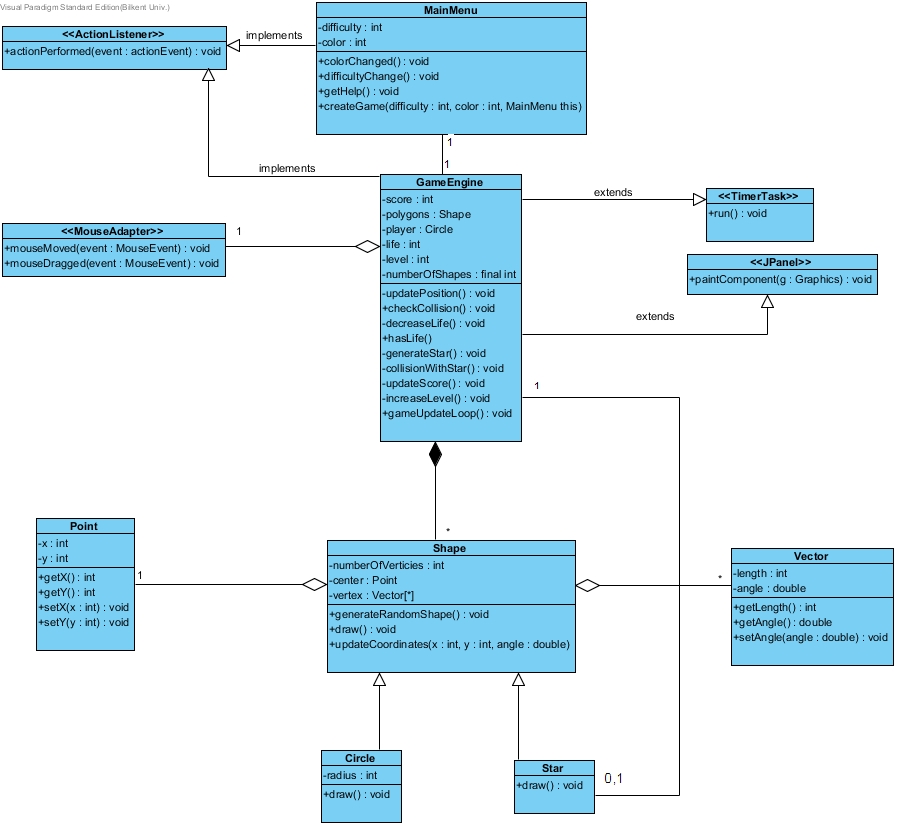
* **Click Area :** It defines areas that can be clicked by user.
* **Main Menü :** The first step that user interact with the game.Most of computer games start with a main menü and allows user to navigate the game with his/her choice.
* **Options :** Allow user to change difficulty of the game and also it gives permission to change color of the main character.

**c)Game Logic**

Game logic is the model of the game. It defines game’s action and interaction between user and game.

* **Bonus:** Specific shape that gives an addition life to the player.
* **Circle:** Specific shape that controlled by player. Circle shrinks when the game level increase.
* **Collision Distance:** Distance between circle and shape. It is the distance of two points: one of them is on the circle’s arc and other one is on the shape’s edge. When this distance become 0, circle and shape collide each other.
* **Game Level:** A game level is a rank that player reachs. Game level increase when player reach a specific score.
* **Life:** It defines the notion of “life”, or the number of “lives” a player has. At the beginning of game, player has specific number of lives and when player lost all his/her lives, game ends.
* **Shape:** Shape objects are actually polygons. Player should try to avoid these shapes in order to increase his/her score.

### 3.1.2. Class Diagram

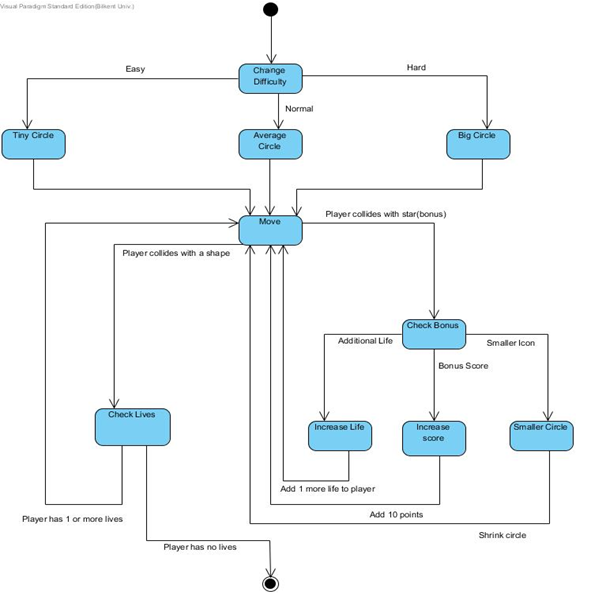


**(Figure 3.1.2 –Shows class diagram)**

## 3.2. Dynamic Models

### 3.2.1. Statechart Diagram

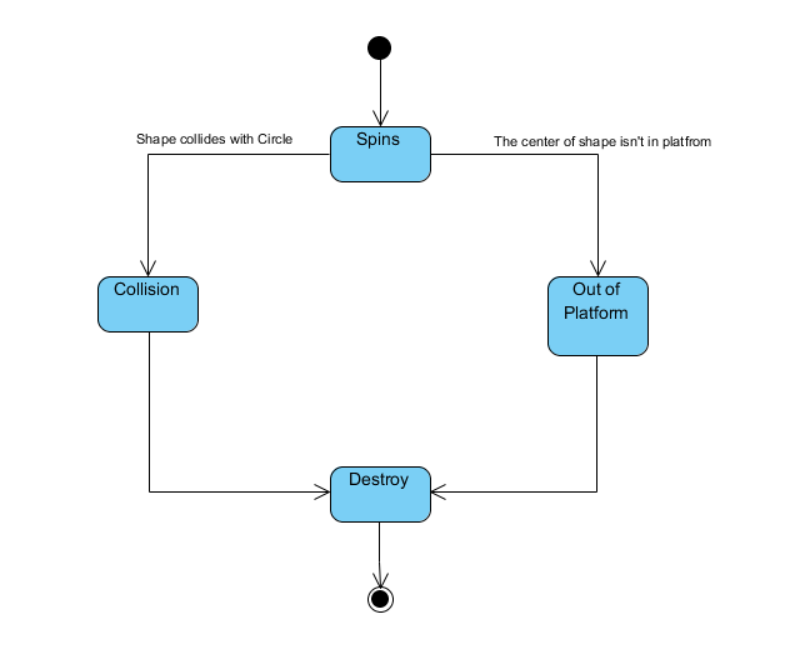
#### 3.2.1.1. Circle State Diagram



**(Figure 3.2.1.2- Shows state diagram of circle which main character)**

Player starts the game with a default diffuculty, which is normal. However, it can be changed depends on the user. Procedure will start with “Difficulty” state and one of the three difficulty choice will be the next state which are “Tiny Circle”, “Average Circle” and “Big Circle”. After the difficulty choice, next state will be “Move” state which define the movement of player on the platform. At this point, circle object either collides with a star or a shape. If user collides with a star, next state will be “Check Bonus” in order to check the type of bonus. There is three type of bonus, one of them is “Additional Life”. It gives an extra life to player. Other one is “Bonus Score” and it gives 10 more points to player. Last type of bonus is “Small Icon”, and it shrinks to Circle of player. After the bonus type state, next state will be the “Move” again and player moves on the platform again. If player lives is 1 or more, next state will be “Move” and player continue to move on platform. However, if player has no life, next state will be end state.

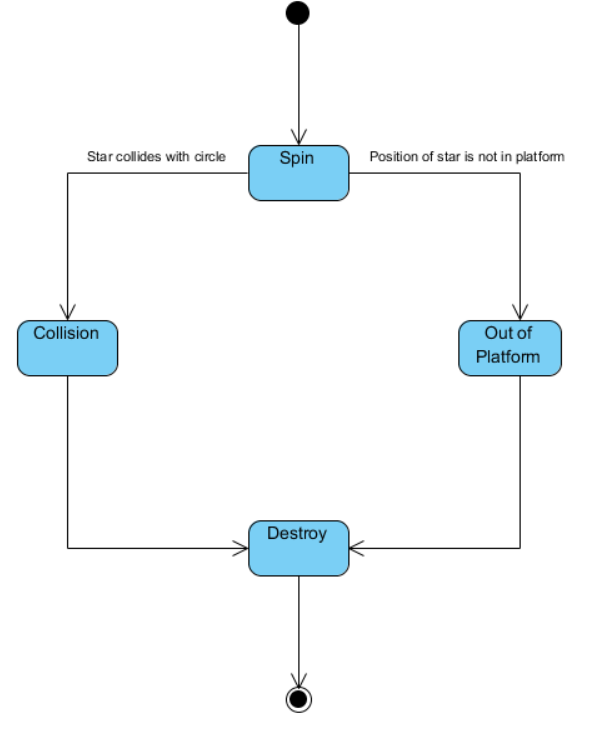
#### 3.2.1.2. Shape State Diagram

****

**(Figure 3.2.1.2 Shows state diagram of shape)**

In the beginning, shape object is in its “Spin” state. It makes state object to spin from top to bottom of the platform. Then, shape object either collides with Circle which belongs to player and goes to “Collision” state, or it will go out of boundaries and goes to “Out of the Platform” state. In both of these states, next state will be the “Destroy” state and object will destroy. After “Destroy” state, end state will be next state and procedure will stop.

#### 3.2.1.3. Star State Diagram



**(Figure 3.2.1.2- Shows state diagram of star shape)**

Star State Diagram is similiary to Shape State Diagram. However, the object in this state diagram will be a star. All states and states direction are same with Star State Diagram.

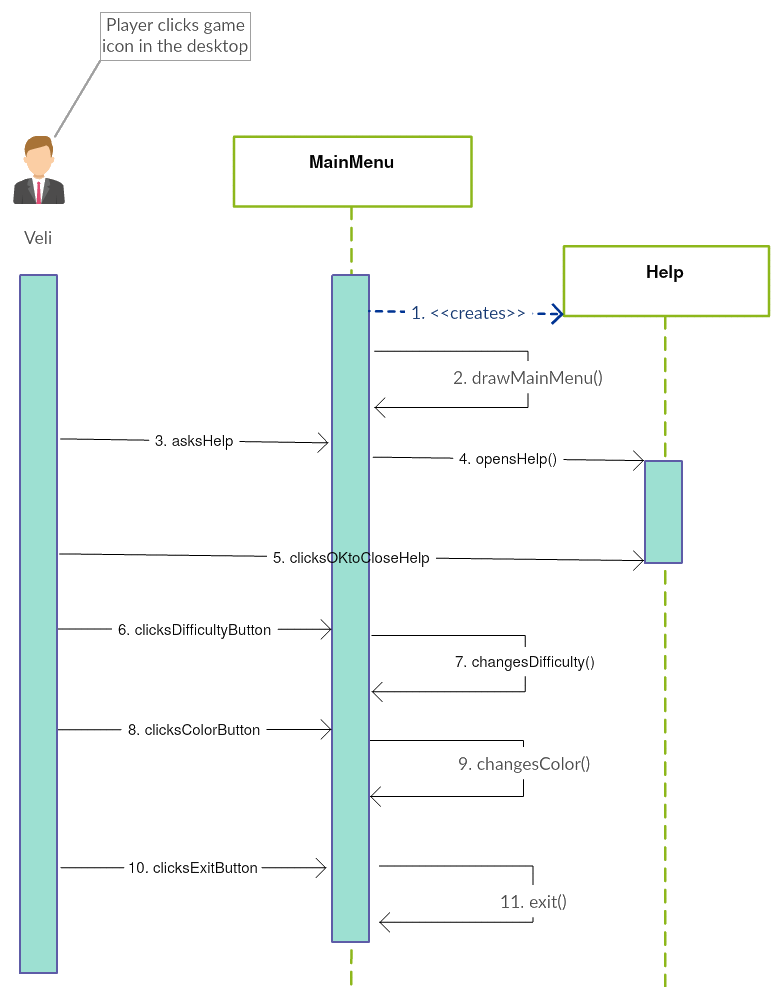
### 3.2.2. Sequence Diagrams

#### 3.2.2.1. Execute Game

Following sequence diagram illustrates the scenario explained below:

**Scenario Name:** Execute Game

**Scenario:** Veli double clicks the game icon in the desktop. Then, he sees the main menu and chooses the “Help” button. Veli reads the instruction which was written there and after understand how to play this game, he select “Ok” buttun from pop-up screen and return to the main menu. Afterward, he chooses “Difficulty” option from main menu. Veli chooses one of the shown difficulty according to his desire and then he returns to the main menu. After Veli adjust difficulty settings, he decided to change to color of his charachter icon. He choose one of the color from main menu and than he decided to exit from the game and choose to “Exit Game” button from the main menu.



**(Figure 3.2.2.1. Shows the sequence diagram which explains execution of game)[2]**

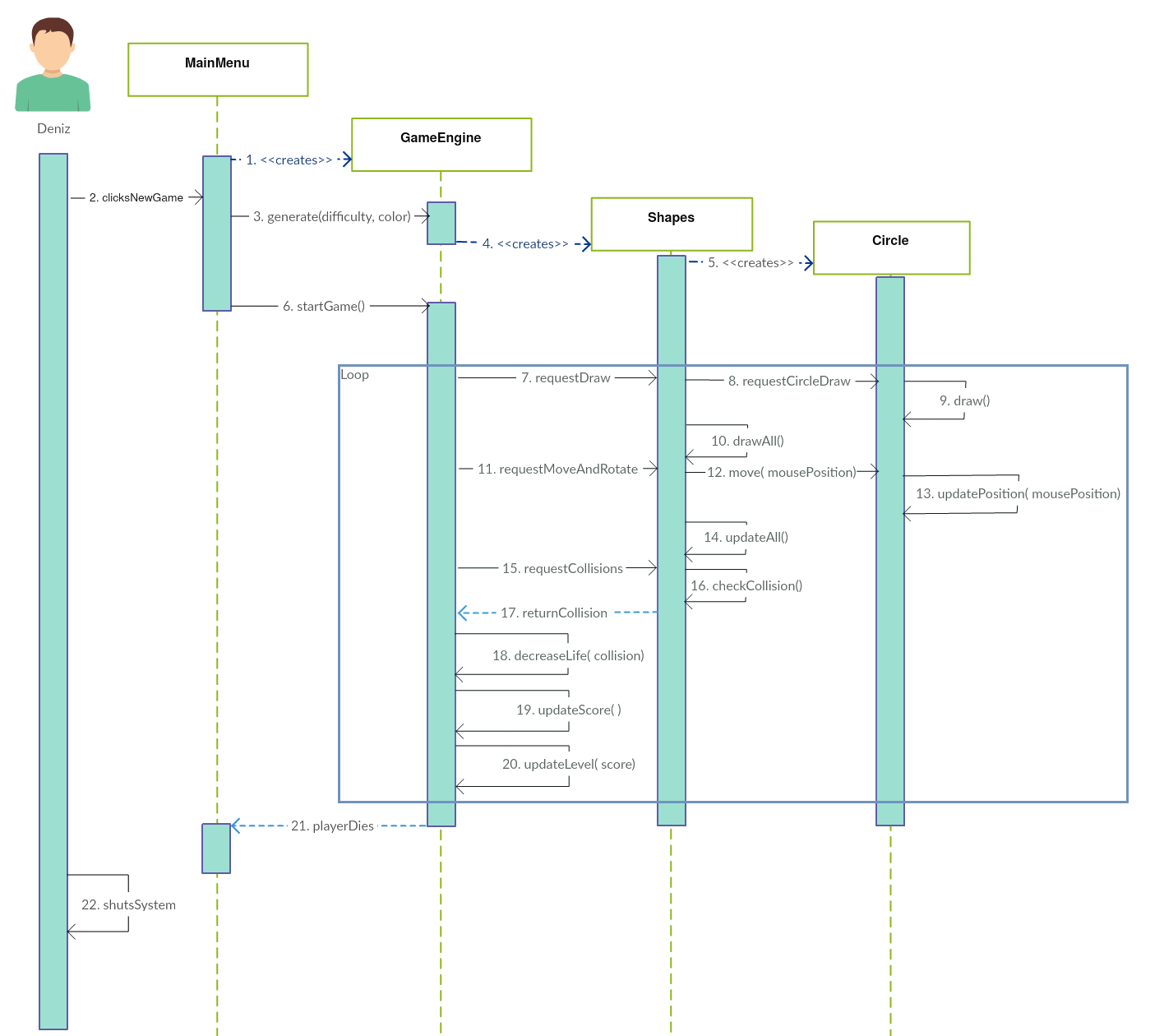
**Description:**After player executes game, player can select help, choose color,difficulty,new game or exit options. In this scenario player choose help,difficulty,color and exit options. While player clicks help button from main menu, new screen will be open and MainMenu creates Helps life cycle. After player reads neccessary information from the help screen, he clicks to “Ok” button from pop-up screen and return to the main menu.

#### 3.2.2.2. Play Game

Following sequence diagram illustrates the scenario explained below:

**Scenario Name:** Play Game

**Scenario:**Deniz decided to choose play game from the main menu. For this scenario we assume that Deniz did same steps with Execute Game scenario but in the end he clicks “Play Game” button rather than “Exit Game” button. After Deniz starts to play game, shapes will start to fall down drom the top of the platform and Deniz avoid from crush this shapes by using mouse. After a while, Deniz hits a shape while he has only one life left and after this crush he died. Than Deniz returns main menu automatically.

**

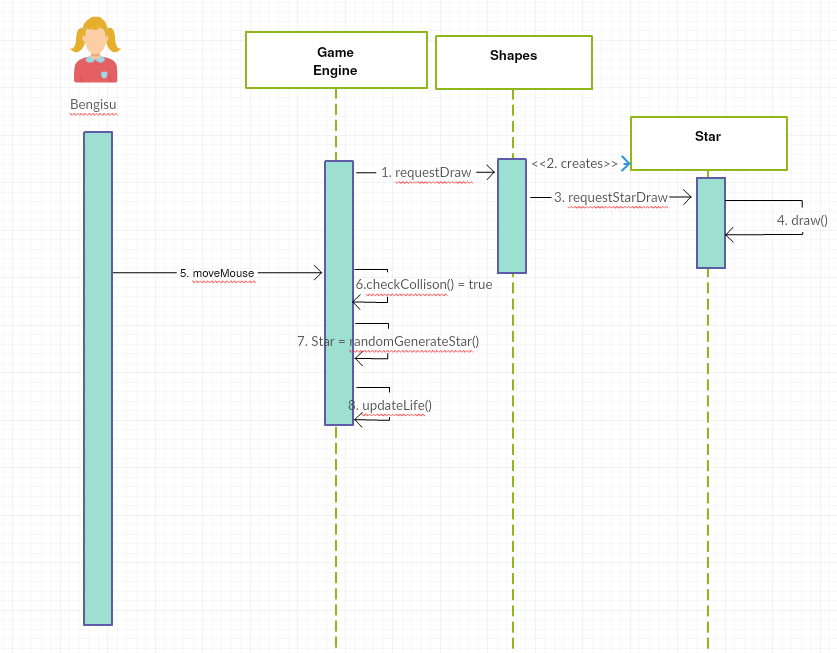
**(Figure 3.2.2.1. Shows the sequence diagram which explains play game case)[2]**

**Description:** After player selects the new game button, the main game loop starts to execute. In every cycle, game loop updates charachter icon and shapes position then checks if any collision happens between charachter icon and falling down shapes. Also in the loop every shapes starts to rotate while they are falling down from the top of the platform. Also in the loop,while a shape is go beyond the boundaries of the platform new shape will be randomly drawn and it starts to fall down from the top of the platform. In addition to these, GameEngine will keep the recordof the current score of the player and if the score bigger than threshhold point, then GameEngine update the level of the game and the shapes starts the fall down faster than before. After game is over, player returns to the main menu.

#### 3.2.2.3. Acquiring Bonusses

**Scenario Name:** Acquiring Bonuses

**Scenario:** Bengisu has already started playing the game. Than suddenly a star shape falling down from the top of the platform and Bengisu captured it. Then Bengisu realise that this is a life bonus and her total life increase by one.

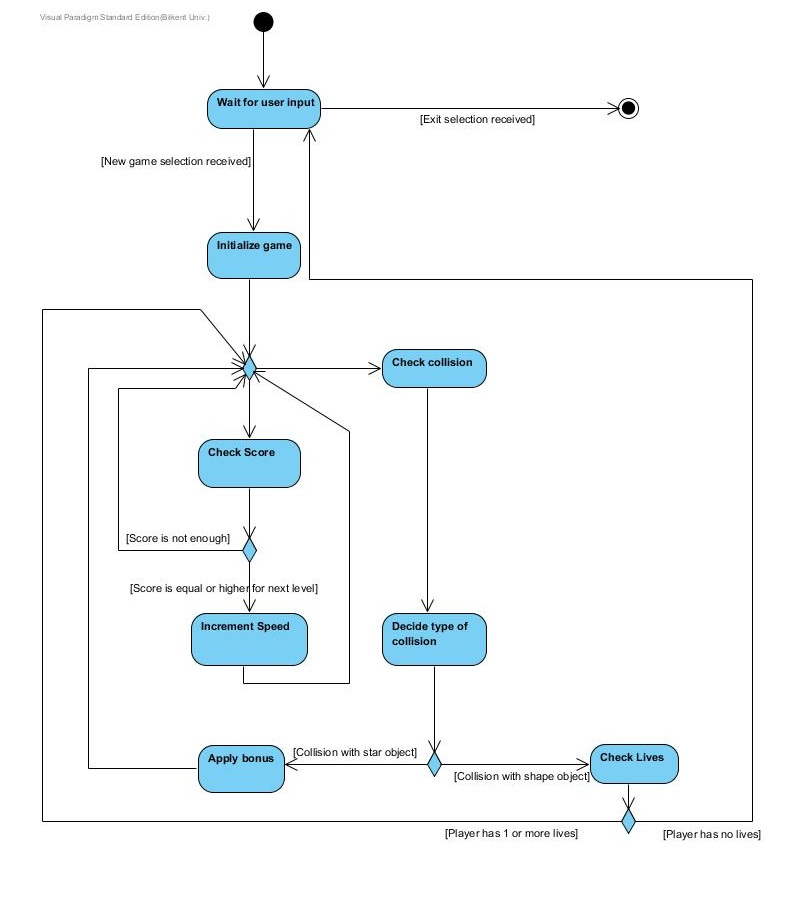
****

**(Figure 3.2.2.1. Shows the sequence diagram which explains getting bonuses)[2]**

**Description:**Shapes will create random polygons and star shape. Star shape can be created same possiblity as other shapes and whenever star shape created from Shapes, a star shape starts to fall down from the top of the platform and if player can catch this star shape than he/she get random bonus which is created by GameEngine. After player gets his bonus, he/she keep playing the game with this extra bonus feature.

### 3.2.3. Activity Diagram

This diagram shows how system maintains gameplay:

****

**(Figure 3.2.3.- Shows activity diagram of the gameplay)**

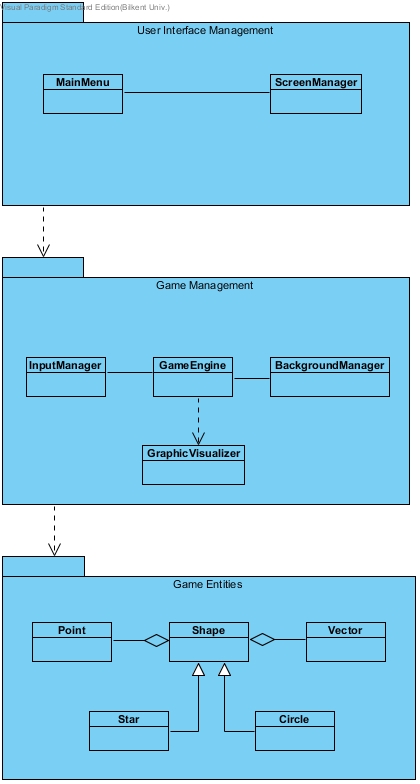
When player selected New Game selection, system initialize game by instantiating game objects and creating game map. When this preparation process is finished, system waits user to interact with the mouse in order to start the game. When shapes start to fall down from above firstly system checks are there any collision between circle and the other shapes. If there is any, system checks type of collision. If type is collision with the star object system increases score of player. After increment of score, system checks score of player in order to decide level up or not .If player’s score is enough system opens the next level and increases speed of game. If score is not enough, player will keep going the same level. Then system checks again to see is there any new collision or not. If system detects a new collision system will tries to categorize type of collision. If type is collision with the other shapes that fall down randomly, system checks life of player that he/she has. If player has life than system allows continuation of the game, otherwise if player has no life, system ends the game.

# 4.Design

## 4.1 Design Goals

* ***Portability* :** Java is flexible language that accepts codes without many changes.When the operating system is changed java is not affected by this kind of changes.Also java is suitable language that convert to codes through android platform with minor changes.
* ***Understanability:***We have aimed that create a game which is easily understandable by the user.Our game is easy to play , for accomplish understandability criterion we created user friendly menu. Difficulty and color settings can be changed only using one button click. Also we put an “Help” button in main menu, if player doesn’t understand the game he/she can get general information about the game-play.
* ***Reliability :*** “Run From Shapes”will be bug-free and non-crashing program because there will not be so many inputs to it and it will not use too much space in stack memory.
* ***Modifiability :*** Since “Run From Shapes” is an object-oriented programmed game, it is always possible to modify existing functions and add new features (new level,new bonuses) to it by simply creating new classes. We aimed well designed subsystem decomposition in order to make changes easily.Each subsystem has relation with at most one level down layer .

## 4.2 Sub-System Decomposition



**( Figure 4.2 –Shows decomposition of the game )**

**Subsystem Decomposition**

We used three main subsystem in our game: User Interface, Game Management and Game Entities.

* **User Interface Management**

User Interface Management subsystem holds necessary elements for user-interface in our game. Game Management consists of graphical objects(panels, frames, buttons) for interacting with user and it also coordinates the transitions between different panels.

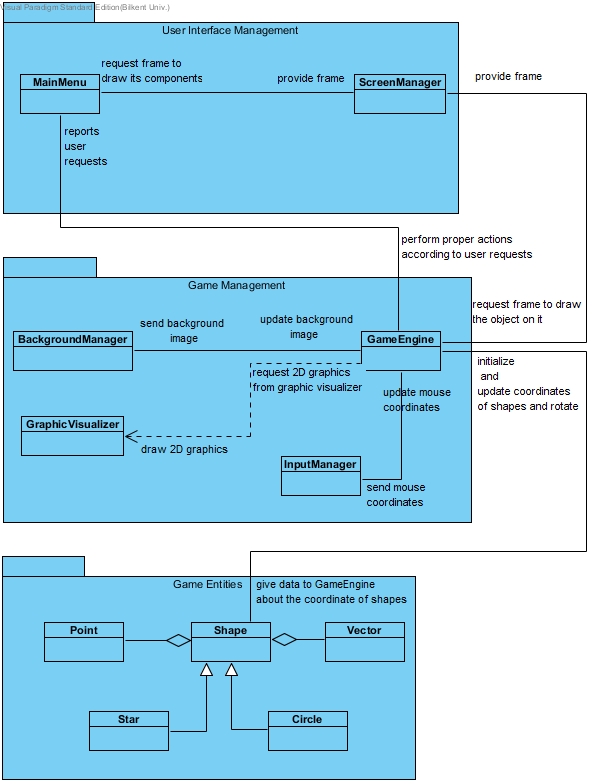
* **Game Management**

Game Management subsystem is responsible from game logic. Also, it is responsible from physics(collision between two shapes, rotation and animation) level/score coordination, input management and map coordination(how many shapes in game, how many bonus etc.).

* **Game Entities**

All entity objects’ classes(Shape, Star, Circle etc.) are holding in Game Entities subsystem. They are created and used by Game Management subsystem.

### 4.2.1 Detailed Sub-System Decomposition



**(Figure 4.2.1- Shows detail of the sub-system decomposition)**

## 4.3 Architectural Patterns

We prefered to use two main design patterns that are Layered Pattern and MVC(model-view-controller) in our game. The relationship between subsystems and MVC architecture is following: User Interface Management is “View”, Game Management is “Controller” and Game Entitites is “Model” part of MVC architecture. We grouped our domain objects into game entities which constitues the model of our system. The domain objects are only controlled and accessed by manager classes that constitues the controller part. Also, we grouped the classes which are responsible for interaction between user and system into User Interface which constitues the view of our system. We chose MVC because it is good a design for games because changes on the interface doesn’t lead to change the model of the system.

In our system decomposition, we seperated our system into three layers: User Interface Management, Game Management and Game Entities. Our top layer is User Interface Management which is responsible for interaction between user and system and it has highest hierarchy because it is not used by any other layer above. The following layer is Game Management which is responsible for game logic. Our bottom layer is Game Entities. Our layers are closed which means a layer can access only to layer below it.

## 4.4 Hardware/Software Mapping

Our game runs on one PC at a time i.e a stand alone system.As ıt can be seen below , there is a deployment diagram of the stand alone system. The programming language that we will use for implementing the core design of our game project is JAVA. We will implement the game helping with Java programming language .Furthermore, Java has large library that helps implementer, for practical usage. Also, Java answers our design goals that are reliable functionality and high performance since its compilers, memory management and multithreading make programs more reliable and faster.

## 4.5 Addressing Key Concerns

### 4.5.1 Persistent Data Management

Game data will be stored in the client hard disk drive, we will not use any database since the data we use in the game needs to be accessed in real-time. Therefore, we will load all the necessary files on to the memory and access those files when the game logic or the rendering system requires. The background images will be stored unencrypted to encourage modifying background images for their personal preference.

### 4.5.2 Access Control Security

Our game will not implement any user authentication system so we will not suppose to have database that stores user type. For the security of our program, the only access to the file system is given to the "GameEngine" which is the main game logic class. Also, we made all critical decision variables in the program, because of that fact it is not supposed to change at any time, constant to assure the security of the information flow. We also decentralized the game logic by delegating basic tasks such as input management. It did not only reduce the complexity of the "GameEngine" but also increase the validity of the code.

### 4.5.3 Global Software Control

In our project, we consider to use event-driven control flow mechanism. In event-driven mechanism, flow of the system is determined by events. There is an event detection part for detecting the changes which occurs in program while it is runing and a part for event handling which will decide what will be done when an event occurs. we decided to use Model-View-Controller, so event-driven mechanism will be the best choice for us because it has separated parts that are control, view and logic parts. Thus, when an event occurs, detection will be made by a part and handling will be made by another part. Furthermore, our project should have decentralized design because we plan to have more than one control object. Decentralized design stands for the distribution of dynamic behavior to objects. Our system will have physics engine because of the collisions and rules engine as control objects, so system should be decentralized to be able to spread responsibility.

### 4.5.4 Boundary Conditions

Game will return to the main menu if all the lives of the player are gone. In case of death, score of the player will be updated and showhed before returning main menu. The game has 5 number of levels and if the user achieves all five levels in the game,he/she will continue to play with maximum speed until he/she will die. If the user quits from the program when he/she returns again score will be set 0 we dont have database system for recording score.As you can see at the below, general boundary conditions have been seperated subtitles for clarifying.

**Initialization**

* User starts the game by opening game file.
* When the game file is opened,main menu shows up by the system.In the main menu there 5 different options which are “Start”, “Help”,”Difficulty”,”Color” and “Exit” .
* If user selects “Start” option, system orients the user to the game directly.
* If user selects “Help” option, pop-up information screen is shown.
* If user selects “Difficulty” option, he/she can set new difficulty for the new game.After setting new difficulty,user can choose “Start” option and than game will be started.

**Termination**

* User has a chance to exit program any time he/she wants, without returning the main menu
* The information about the system that are kept in memory is cleaned up so that memory

should be emptied and so the system could start from the beginning next time.

* If there is any high score in incomplete game, they can not be saved before termination. In this case,after re-opening the game, user will start the game from the zero.

# 5. Conclusion

We created our analysis and design report in order to design and implement a game, called “Run from Shapes”. Our report consists of three main parts. First part has focused on requirement specification. Second part has focused on system model and third part has focused on the design of our project.

For requirement specification part, we tried to examine all possible requirements which a player could perform with the mouse controlling game. In our project design, we plan to fulfill all of these requirements. After deciding which requirements which a player needs, we were ready for planning our system model.Our system model, which is second main part of our analysis and design report, consist of four sections these are:

1.Use case model

2. Dynamic models

3. Class model

4. User interface

For deciding our use cases that we have, we spent some time on thinking about them in order to identify uses we should indicate for our project. At this point, we tried to specify requirements and actions that are actual use cases.

Our Dynamic model includes sequence diagrams, state diagrams and activity diagram. We tried to show possible actions that will constitue the crucial parts of our game and interactions between player and the System by sequence diagrams. We tried to identify all possible states in our state diagrams. Our activity diagram indicates our game play basically. In our class diagram, we tried to form a perfect class model since we know that in implementation and design process this model would constitute a crucial part. We spent so much time about thinking possible classes and connections between them.

Fourth part of our analysis report is user interface and navigational path diagram. We tried to keep user interface mock-ups simpler and fancy as much as we can, since we plan to have a user friendly interface. And we created a navigational path with using our use cases.

In design chapter of this report, firstly, we decided our design goals. Then, we merged our classes into subsystems in order to their relationship and we indicated our subsystem decomposition. In architectural patterns topic, we explained the architectural styles that we used in this project. Then we explained hardware/software mapping and we addressed our key concerns in other topics.

To sum up we tried to create a perfect analysis and design report since it will guide us in our implementation process. We do not want to deal with problems in future, so this is why we took serious on analysis and design report.

# 6. Reference

[1] <https://www.gliffy.com/go/> . 14.October.15

[2] <https://creately.com/>. 21.October.15