CS319 - OBJECT ORIENTED SOFTWARE ENGINEERING
SPRING 2015

Analysis Report

Ali Kaviş - Burak Kantarcı - İnci Boduroğlu - Uğur Akkurt

GROUP 4 | SECTION 2

23.03.2015

INTRODUCTION

As a group project for CS 319 course, our group has decided to develop a virtual arcade game of our own design. In this project, we will be following object-oriented software engineering techniques and guidelines to design and construct the game. Our project Hockey Champs is a fusion of a virtual hockey game and a brick breaking game. In this project, our goal is to successfully combine these two types and produce a game that functions properly and includes main functionalities of both.

The game offers various functionalities and features that are adopted from both of the game types. The game is played by one user at a time versus AI of the game. The aim is to score by controlling the bar and win against the AI by fulfilling the requirements of each map.

Hockey Champs was chosen because we think it fits the CS 319 project requirements perfectly. There is significant amount of interacting objects and classes in order to develop an understanding of Object Oriented software design. Since the proposed system resembles an actual life concept and has aspects that can be expanded on, both initializing a system as-is and doing visionary work is possible.

The system will be developed in 3 main steps: analysis, design and implementation. This report embrace the first part of the project, namely analysis. First part of the report is requirements analysis part which corresponds to the initial steps of the design. This part includes functional and non-functional requirements with detailed explanations of their subparts. Second part is the system analysis which focuses on use-case models, object models and dynamic behavior of classes.

Scope of the system

The game will be designed in accordance with object-oriented software design guidelines using Java as the programming language since it offers practical and useful GUI libraries that come in handy while implementing graphical and visual elements . The main goal is to achieve well designed, reusable and unique software according to the fundamentals of the Object Oriented Programming. Game will be played by user against computer with a basic artificial intelligence developed by our team.

Objectives and success criteria of the project

As mentioned above, our main objective is to develop well designed, stable, unique and reusable OOP Game. Some success criteria of these goals are following:

- Identifying requirements for developing an interactive game and elaborating on them
- Doing extensive research about similar games, learn about their fundamental features and make our project unique.
- Creating clear and realistic game scenarios and work flow beforehand.
- By using our research results and elaboration on work flow, producing a comprehensible and robust design for our system.
- Understanding the OOP fundamentals and design software accordingly.
- Designing a consistent system that abides by OOP fundamentals which properly models the real-life objects, interactions and events
- Understanding and applying specific design patterns that are most appropriate to our respective system.
- Adopting MVC design pattern that handles interactions between different parts of the system such as view and model packages in an orderly fashion so that system function smoothly.
- Obeying the due dates designated by the team

PROPOSED SYSTEM

Overview

BASIC GAMEPLAY

Hockey Champs is a virtual arcade game with one player playing versus the game's AI. The aim of the game is to score a particular amount of goals before the AI or score more than the AI before the time runs out. In different maps, the victory requirements will vary. User plays the game by using the arrow keys and control keys can be re-assigned according to user's preferences.. The control keys move the user's bar so that user can block the puck from getting into his/her goal and can give direction to it to break some obstacle or collect power-ups. Hockey Champs will have various maps with various difficulties.

GAME OBJECTS

Every map will have their own locations for the game objects, meaning configuration of obstacles and randomly appearing coins will vary from map to map. There are 4 main types of game objects: bar, puck, coin and block. Each game object inherits other objects of their type which possess features and attributes.

1. PUCK

The puck can move within the arena boundaries. The user and AI use their respective bars to bounce the puck towards the other players goal and when the puck goes into the goal of a player, the other player scores and their score is increased by one. The puck's movements are controlled by the system using its algorithms but depending on the movement direction and speed of bars, when puck gets into contact with bars, its movement direction will adjust in a particular way that its speed is kept constant. There is only one type of pucks but they come in different textures and colors as shown in figures.

Figure 1 - Puck Types

2. COINS

Coins are the objects that are generated by the system at random places of the map. Coins have two different types: power-ups and penalties. When the puck collides with any coin appearing on the arena, they immediately disappear and effects of the coin is reflected on the game and players by the system and related game objects are updated accordingly. There are 10 coins at total that have different effects on the gameplay. 5 of them are power-ups and the remaining 5 are penalties. System knows where each coin can appear and chooses a random place among them if there is not any other coin occupying that location for the time being.

Power-up coins can shrink the AI's bar for a certain time period, decelerate AI's bar for a certain time period, disappear the AI's bar for a certain time period, accelerate user's bar for certain time period and increase the score of the user. There are 5 distinct coins for these purposes as shown in the figures.

Penalty coins can shrink the user's bar for a certain time period, decelerate the user's bar for a certain time period, make the user's bar invisible for a certain time period and decrease the user's score permanently. The 4 different types of coins that perform these actions are shown in the figures.

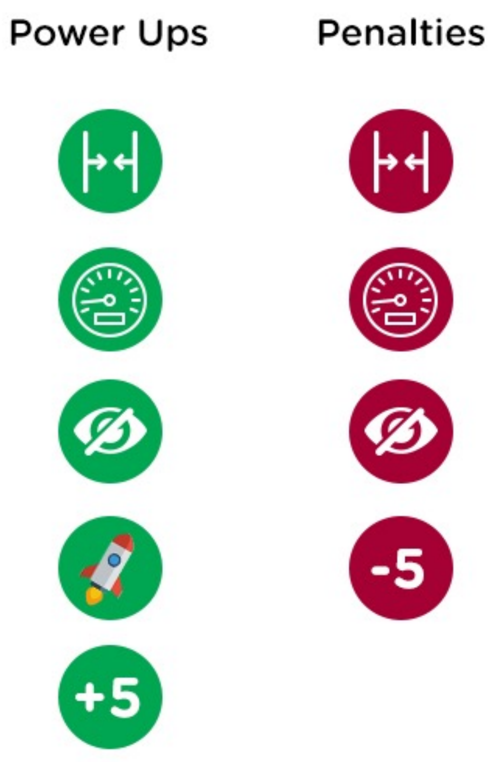


Figure 2 - Coin Types

3. BLOCKS

Another game object type is blocks which the puck can also hit. Some blocks can shatter upon impact and some bounce the ball back within the principles of elastic collision. In fact, all collisions are assumed to be elastic in order to foster a maintainable and consistent gameplay. The blocks are created as the game is launched and are all concrete. Breakable blocks disappear upon certain number of collisions, but concrete and indestructible ones remain as long as the game proceeds. User can eliminate some blocks to ease gameplay and score with less effort. The system does not create new blocks as the game progresses as it creates coins. All the objects locations are predetermined in the maps properties. While the blocks locations are fixed, the coins locations can vary between predetermined possible locations, in a randomly fashion.

There are 2 main block as indicated above: breakable and solid but depending on the strength of the breakable ones we have 3 levels of breakable blocks which all have distinct textures and disappear on various number of collisions. All types of blocks are described and shown in the figure.

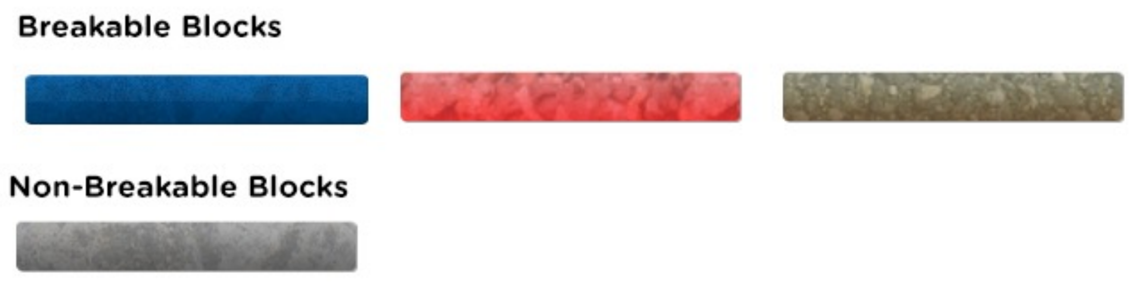


Figure 3 - Block Type

4. *BAR*

Final type of game object is the bar. There is only a single type of bar for both user and the AI. User can change the texture and color of the bar, which are demonstrated in the figure.

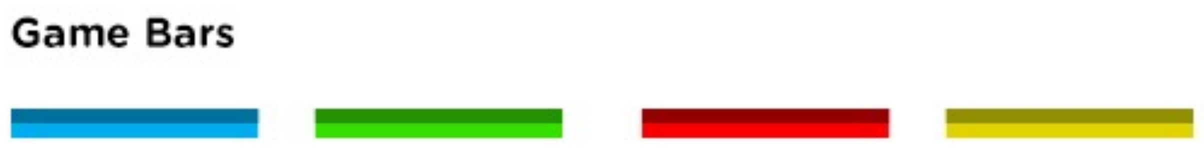


Figure 4 - Bar Types

MAPS

Each profile comes with two different maps that are available for playing. Maps are initiated with a certain configuration of blocks and it is unique among all maps. For each map, there are designated places for coins to appear through the game and system generated them randomly. There are other maps that requires to be unlocked before being able to play them. These maps can be unlocked via market screen in which user pays a certain amount of points to unlock them. Example configuration for a map is given in the figure.

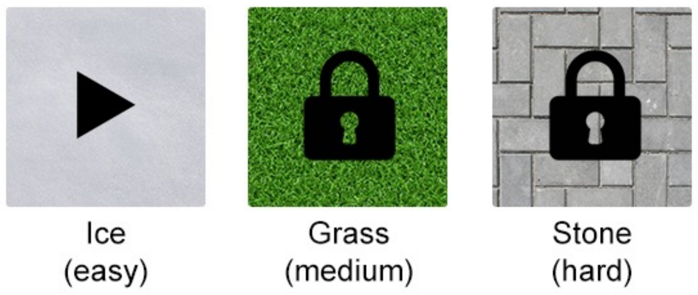


Figure 5 - Map Selection

SETTINGS

The settings of the game can be changed in the Settings Menu. The User can change the control keys, the volume, background of the game and; the bar and the puck's color and texture from one of the pre-determined one's. Any changes that are saved by the user are stored in user's profile and the user play with the latest preferences in the following games.

VICTORY CONDITIONS

In order to win the game, the user has to score certain number of goals the first. When the User scores certain number goals the game terminates and the total score of the user is saved on high score table of the corresponding map. Similarly, if the user scores more than the AI when the time runs out i time-attack mode, user's total score is posted on high score table.

Functional Requirements

Hockey Champs is a Java client-based, arcade game with two players, one user and one AI. Hockey Champs combines fundamental features of air hockey and brick breaking games. The system uses specifically developed algorithms to react to the changes in the game.

Users can launch the game using the desktop shortcut to the executable. After launching, users can choose to signin or sign up. New users must sign up to play the game, while users with profile can proceed to the game menu.

The system includes a user authorization procedure where users must identify themselves using a login name and password. Users who have not created a game profile must first sign up and create a unique profile consisting of an id and password because guest user's are not allowed by the system due to security purposes. ID's that are provided by the user are checked by the system. System checks whether the provided ID already exist in the system or not. Unique IDs are accepted and system successfully creates a new profile.

Users that have created their account are granted full access to the system. Users must enter their respective id's and password's to login. System informs the user by displaying whether the sign-in is successful or not.

There is only one type of user in the game, so all authorized user can use whole functionalities of the game. Any user can play the game, unlock new maps, set preferences for game visuals, sound and game controls, learn more about extensive features of the game, discover about our team and exit the game whenever they want.

Users can play any of the maps that are available to their profile. User should select a map to start the game. After user chooses a map and starts the game, system prepares the arena for the user. Users cannot play unavailable maps as long as they do not unlock them. System checks the user profile and displays available maps.

Users will have a total Score point, which is a sum of all the scores user gets after each game. This total score will be shown on the top of the screen as long as the user is signed in, which can be spent to unlock new maps from the main menu if he/she has enough score to unlock. User can only make points during a match. User can score a goal, collect certain coins that give quick points, break particular blocks and win a match to earn points.

User can use his/her points to unlock new maps that are available in in-game market. User can see the prices of each map with brief information about each map that indicates the difficulty level and arena type. User must be able to unlock a new map as long as he/she has enough points to unlock a map. User cannot unlock maps if he/she has insufficient funds. System checks every attempt by the user to unlock a map and display a message that either indicates a successful purchase or failure to unlock a map. When user unlocks a map, system deducts the respective amount from user's profile and automatically opens the map in play game mode.

User can change the settings of his/her profile when he/she is not playing any match. User can access the Settings menu from main menu. System displays the most recent form of the system preferences for each particular profile. User can change the volume between maximum and minimum values, mute or unmute the in-game sounds, adjust control keys but control keys should be chosen among provided keys, set the image that appears on the puck to on of given images and change the color of the bar. User should save any modification before leaving the settings menu, otherwise system does not apply any changes to the profile.

The game play as seen in the mockups (2.4.5) consists of different objects with different attributes. The power-ups and the obstacles are placed on a map randomly. User cannot change neither the orientation nor the number of obstacles on a given map. System knows the locations where a power-up can show up on the arena. The user's goal is to block the puck and give it the appropriate direction to score and win the game. While playing the game, the user will use the arrow keys to move their bar right and left. User should not be able to move their bar up and down for the sake of simplicity. The user can see their in-game score on the screen. Total score is update after each game. If their score can make it to the high scores table the user can save their score with their username.

The user can also choose to view the high scores or credits from the main menu screen. User should be able to learn about features of the game from Help screen. User can access help screen from main menu.

Nonfunctional requirements

USABILITY

- Users who are familiar with arcade games like DX-Ball or Air Hockey will not be needing detailed explanations in Help mode.
- User should be able to create a unique profile by providing a unique name and password using the text fields provided for them.
- User can easily access a profile by typing a name and the corresponding password into specified fields.
- User should be able to access the main menu automatically by logging in to an existing profile, without performing any further action.
- Any user should be able to start a new game with less than 10 mouse clicks.
- User can adjust background, sound and bar color, all at once, via Options screen.
- Users with enough points can unlock new maps with 2 mouse clicks on Market screen.
- Gameplay is sufficiently straightforward so that users can get used to it quickly.
- User can learn about any feature of the game from Help screen.

RELIABILITY

- If system crashes in the middle of a task, which means during a game, no prior progress is lost due to this inconvenience.
- When a game is completed without any failure, system saves progress for the specific profile.

- Any unauthorized access to the system and data is not allowed, that is no user can access the main menu without logging in.

PERFORMANCE

- System should respond to user’s command I no longer than 50 milliseconds.
- Graphical computations should be executed with an onboard GPU.
- System should run properly in any display supporting a 1100x720 resolution.

SUPPORTABILITY

- The system will be able to be developed, such new attributes and functionalities can be added without changing the core structure of the system. Changes imposed by new technologies and bug fixes will be easy to implement.
- The system will support English and Turkish languages.

IMPLEMENTATION

- The system must be implemented in Java because it is a widely used programming language that is perfectly suitable for object-oriented software implementation which is needed for this hockey game which includes a lot of objects and interactions among them.
- Any system that has Java libraries installed in itself will be able to run the game executable.

- The system must be implemented in Eclipse IDE since it has functionalities that ease the coding process and it enables users to connect to a project on Google Code and manipulate files on a Google Code page so that project members can work on a given document or file simultaneously.

INTERFACE

- User will be able to play game easily on the screen. The components of the game such as bars, puck types, blocks etc. are going to be easily determined and well designed. In addition, because the game will be played on the screen by using keyboard and mouse, the graphical user interface components such as buttons will be easily clickable and recognizable.

LEGAL

- The project should be licensed under Apache 2.0 open-source license, which is one of the most widely used license by open-source projects.

System models

SCENARIOS

Scenario Name	UserHitsThePuck
Participating Actor Instances	**Berkcan: User**
Flow of Events	Berkcan wants to hit the puck and bounce it back to protect his goal.
	Berkcan presses the movement key that moves the user's bar to the right
	System responds to the input and moves the bar to the right, as long as Berkcan presses on the key.
	After Berkcan is presses the right key and moves the bar, puck and bar collide.
	System detects the collision and changes the puck direction accordingly.
	Puck bounces back within the rules of the collision algorithm and drifts away from Berkcan's goal, system draws the final states of the objects after collision.

Scenario Name	**UserBarAccelerateCollection**
Participating Actor Instances	**Berkcan: User**
Flow of Events	After Berkcan hits the ball while moving to the left and changes the direction of the ball to collect the userBarAccelerate coin, system detects the collision and handles it.
	System sets the new velocity of the puck after collision and updates the view.
	Puck drifts towards the coin with each periodical update of the system and reaches the coin within a certain number of update loop.
	System detects the collision of the puck and the coin, analyzes the type of coin, which is userBarAccelerate power-up, and updates the userBar speed accordingly.
	After adjusting the speed of the bar, system initializes a timer that keeps how long the power-up will be active on the user bar.

Scenario Name	**BreakBrick**
Participating Actor Instances	**Berkcan: User**
Flow of Events	Berkcan catches the puck with his bar by moving it.
	When he hits the puck, it drifts towards the brick with each periodical update of the system and reaches the brick within a certain number of update loop.
	System detects that collision occurs between the puck and the respective breakable brick.
	After collision puck breaks the brick that it get into contact with.
	System handles the collision according elastic collision principles and puck attains its new vector keeps on moving.

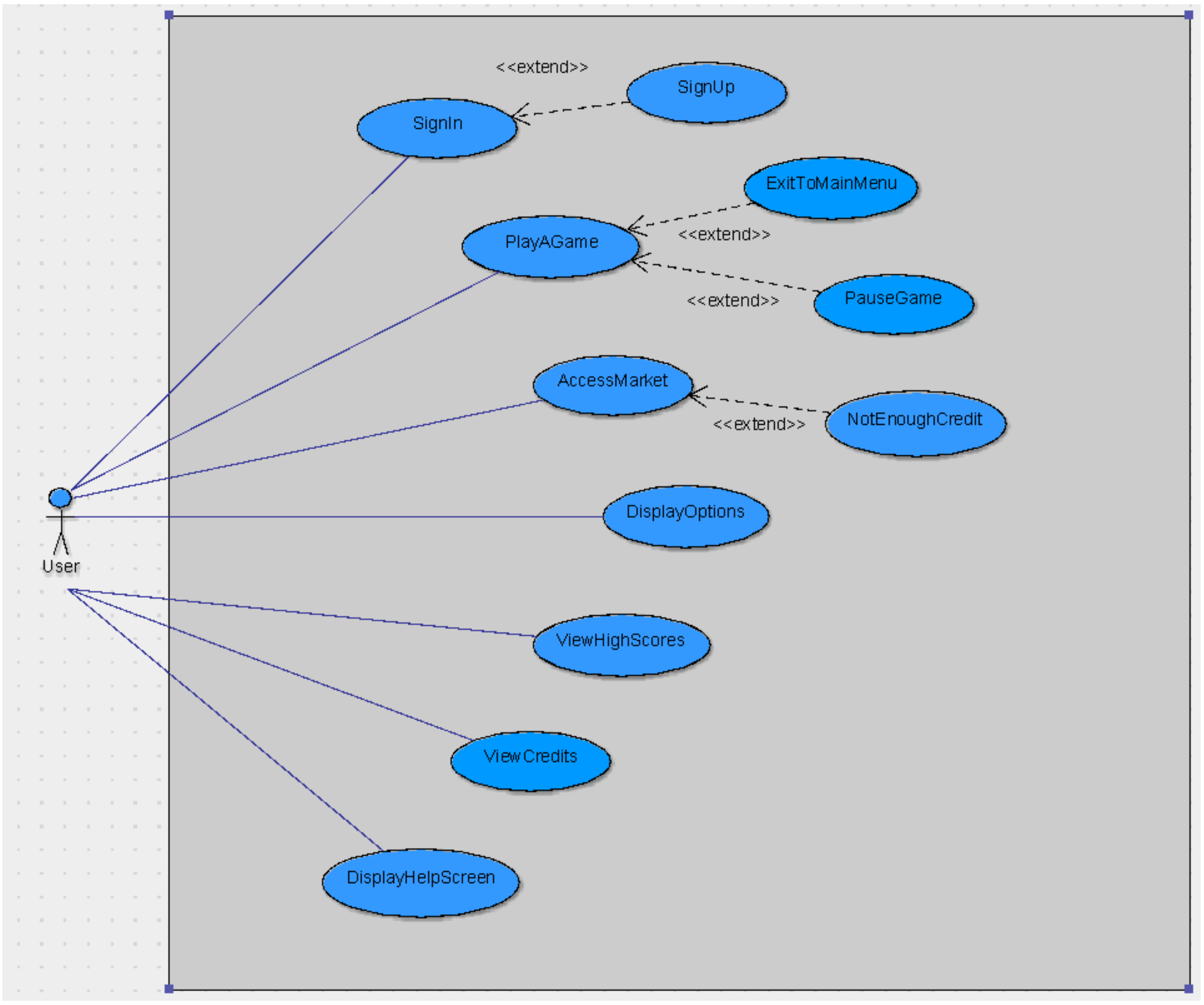
Scenario Name	SuccessfulSignUp
Participating Actor Instances	**Berkcan: User**
Flow of Events	Berkcan wants to play the game and he tries do so without providing any username or password.
	System notifies him that he should create a profile or enter a correct combination for signing in.
	Berkcan clicks sign up button when he is in the opening screen of the game.
	Berkcan types a username and password that he prefers and clicks ok button.
	He sees a window dialog saying that his choice of ID is unique and confirming that his profile creation process is complete.
	He closes the dialog by clicking OK button.
	System displays the opening screen again with text fields for username and password.
	Berkcan enters the profile information that he recently created and hits enter.
	System display a confirmation message and Berkcan gains access to main game menu.

Scenario Name	SuccessfulSignIn
Participating Actor Instances	**Berkcan: User**
Flow of Events	Berkcan clicks sign in button when he is in the sign in-up menu.
	Berkcan sees the sign in panel and the form that needs to be filled in to sign in.
	He types his username and password that he set before for signing up.
	He sees a window dialog confirming that his his/her signing up process is complete.
	He closes the window dialog by either clicking ok or x mark.
	He now sees the main menu.

Scenario Name	**NewMapPurchase**
Participating Actor Instances	**Berkcan: User**
Flow of Events	Berkcan sees the main screen in his profile. He clicks on the button that bring up the market screen.
	System displays the market screen. By validating the information in Berkcan's account, system displays the maps that are already purchased in grayscale blur and the maps that are available for purchase with colored images,
	Berkcan accesses the market screen to purchase a new map to play.
	Berkcan has collected enough points to purchase and unlock any map listed on the market.
	Berkcan decides to purchase the third map and selects the respective field. Then, Berkcan presses on "Purchase" button to buy the chosen map.
	System validates that Berkcan has sufficient credits to unlock the map, updates Berkcan's account by adding the third map to purchased-map list, deducts the respective amount and blurs the selected map.

Scenario Name	**PreferenceChanging**
Participating Actor Instances	**Berkcan: User**
Flow of Events	Berkcan logs into his profile and wants to change the settings of his profile from main menu. He goes to settings menu to make the necessary changes.
	Berkcan mutes the sound to eliminate any in-game sound.
	He also changes the background and chooses one of the options provided on the screen
	To change the bar color, he goes to the list of available bar colors and textures and selects one of them.
	Berkcan saves the changes before leaving the Settings Screen to apply the changes he made on preferences.
	System updates the preferences associated with Berkcan's profile and it makes them available in the next game Berkcan plays.

USE CASE MODEL



Scenario Name	SignIn
Participating Actor Instances	**Initiated by User**
Flow of Events	1. User launches the game.
	2. System prepares the opening screen and displays text fields for username and password, together with sign in and sign up options.
	3. User fills the text fields by typing his/her username and corresponding password.
	4. User clicks the ok button after completely filling in the text fields.
	5. The system receives the username&password combination and validates whether the given username and password match. If the given information of the user exists in the system files, the system permits the sign in and uploads user's current unlocked maps, total score, preferences and informs user by showing the dialog window that the sign in process is complete.
	6. User closes the dialog window by either clicking ok button or X mark.
	7. System prepares the main screen of the game for the profile owner.
Entry Condition	**Users enters a username&password combination and clicks on sign in button.**
Exit Condition	**User signs in with valid username and password.**
Quality Requirements	**The systems response time should be 1/10 seconds**

Scenario Name	SignIn
Participating Actor Instances	**Initiated by User**
Flow of Events	1. System prepares the signup screen with username and password text fields, together with an ok button.
	2. User fills the text fields by typing a username and password that he/she chooses
	3. User clicks the OK button and sends data to validation process.
	4. System checks whether the typed username is unique. If it is not unique, system opens a dialog window that indicates username already exist in the system and the use case goes back to 3rd step.
	5. System checks whether the typed password is valid. If it is not, system opens a dialog window showing that password is not valid and the use case goes back to 3rd step.
	6. If the typed username and password are both valid, system completes the signup process and updates the respective system files by creating a new profile and opens a windows dialog showing that sign up process is complete.
	7. User closes the dialog window by clicking ok button and comes back to the opening screen where he/she can choose to sign in or sign up.
Entry Condition	**User launches the game and clicks on sign up button.**
Exit Condition	**User creates the account successfully and returns back to opening screen.**
Quality Requirements	**The systems response time should be 1/10 seconds**

Scenario Name	PlayGame
Participating Actor Instances	**Initiated by User**
Flow of Events	1. System shows User the Main Menu
	2. User clicks Play Game option
	3. System shows the available maps. Maps are determined according to their difficulty. The screen shows default maps and the maps user can purchase from the market.
	4. User picks a map by clicking on the maps image with their mouse.
	5. System starts the game. The game map is launched with maps specifications. The objects on the map is determined by the difficulty of the map. Game starts with 0 points. If the User has assigned preferences, the system starts the game according to pre-assigned options. If not, system launches the game with default options.
	6. System launches the puck from its starting point with a fixed starting velocity and accelerates using its algorithms.
	7. User presses right and/or left arrow keys to play the game. If the user has assigned another keys (from the Setting Menu), these keys are used to play the game.
	8. System responds to the respective arrow key by moving the users bar left and right. When the puck and a bar collides system uses its algorithms to bounce the puck. The bars movement at the instance of collision effects the pucks movement by either increasing or decreasing its movement on the x-axis by keeping the total speed constant.
	9. User collides the puck with a coin.
	10. System updates the Users or the AIs objects according to the coins properties. Power-up coins can destroy, shrink or decrease the speed of the AIs bar; Penalty coins can shrink, vanish or decrease the speed of the Users bar.
	11. a. User couldn’t block the puck with their bar meaning that the AI scores a goal.
	12. System increases the AIs score by one and initiates the game from (5). If the AIs score has reached 7 the game terminates and shows the user their score.
	11. b. User scores a goal when the AI can’t block the puck with its bar.

	12. System increases users score by one and initiates the game from (5). If the AIso score reached 7 the game terminates and shows the user their score
Entry Condition	**User launches the game.**
Exit Condition	**User clicks the back button.**
Quality Requirements	**The systems response time should be 1/10 seconds**

Scenario Name	**ViewHighScores**
Participating Actor Instances	**Initiated by User**
Flow of Events	1. User runs the game and successfully signs in to a profile.
	2. System brings main menu screen.
	3. User clicks High Scores option from the main menu.
	4. System shows High Score screen.
	5. User sees the list of the high scores. User will be able to see results of the games. High scores will be listed according to score difference between AI and also the time that user had played to win in that particular game.
	6. User clicks Back button to exit to main menu screen.
	7. System opens Main Menu screen.
Entry Condition	**User launches the game.**
Exit Condition	**User clicks the back button.**
Quality Requirements	**The systems response time should be 1/10 seconds**

Scenario Name	DisplayOptions
Participating Actor Instances	**Initiated by User**
Flow of Events	1. System brings main menu screen when user runs the game
	2. User clicks Settings option from the main menu.
	3. System shows Settings screen.
	4. User selects the background tab.
	5. System shows background selections. User will be able to change background color, buck color and general style of the game according to their wishes. If user selects background tab there will be several background selections which determined by the developers. Users will not be able to upload their own images.
	6. User selects the puck tab.
	5. System shows puck selections. User will be able to change puck color according to their wishes. If user selects puck tab there will be several puck style selections which determined by the developers. Users will not be able to upload their own images.
	8. User clicks save button and system updates the display options according to the Users preferences.
	9. User clicks Back button
	10. System opens Main Menu screen.
Entry Condition	**User launches the game.**
Exit Condition	**User clicks the back button.**
Quality Requirements	**The systems response time should be 1/10 seconds**

Scenario Name	PauseGame
Participating Actor Instances	Initiated by User
Flow of Events	1. System brings main menu screen when user runs the game
	2. User clicks Play Game and initiates the preferences of the game.
	3. User starts to play game
	4. User selects the pause icon in the game screen.
	5. System pauses game for both user and AI. User will be able to pause the game until the system closes. System will not be able to save game from where it paused by user.
	6. User clicks Continue button
	7. System continues the game. If user wants to play game from where it paused, he/or she can press the continue button and play.
Entry Condition	**This use case extends the PlayAGame use case. It is initiated whenever the game is stopped due to a user's interference.**
Exit Condition	**User clicks the back button.**
Quality Requirements	**The systems response time should be 1/10 seconds**

Scenario Name	NotEnoughCredit
Participating Actor Instances	Initiated by User
Flow of Events	1. System brings main menu screen when user runs the game
	2. User clicks Maps.
	3. System brings the available maps to the screen. There will be several map selections on the screen and user will be able to purchase and use available maps.
	4. User selects one of the maps on the screen.

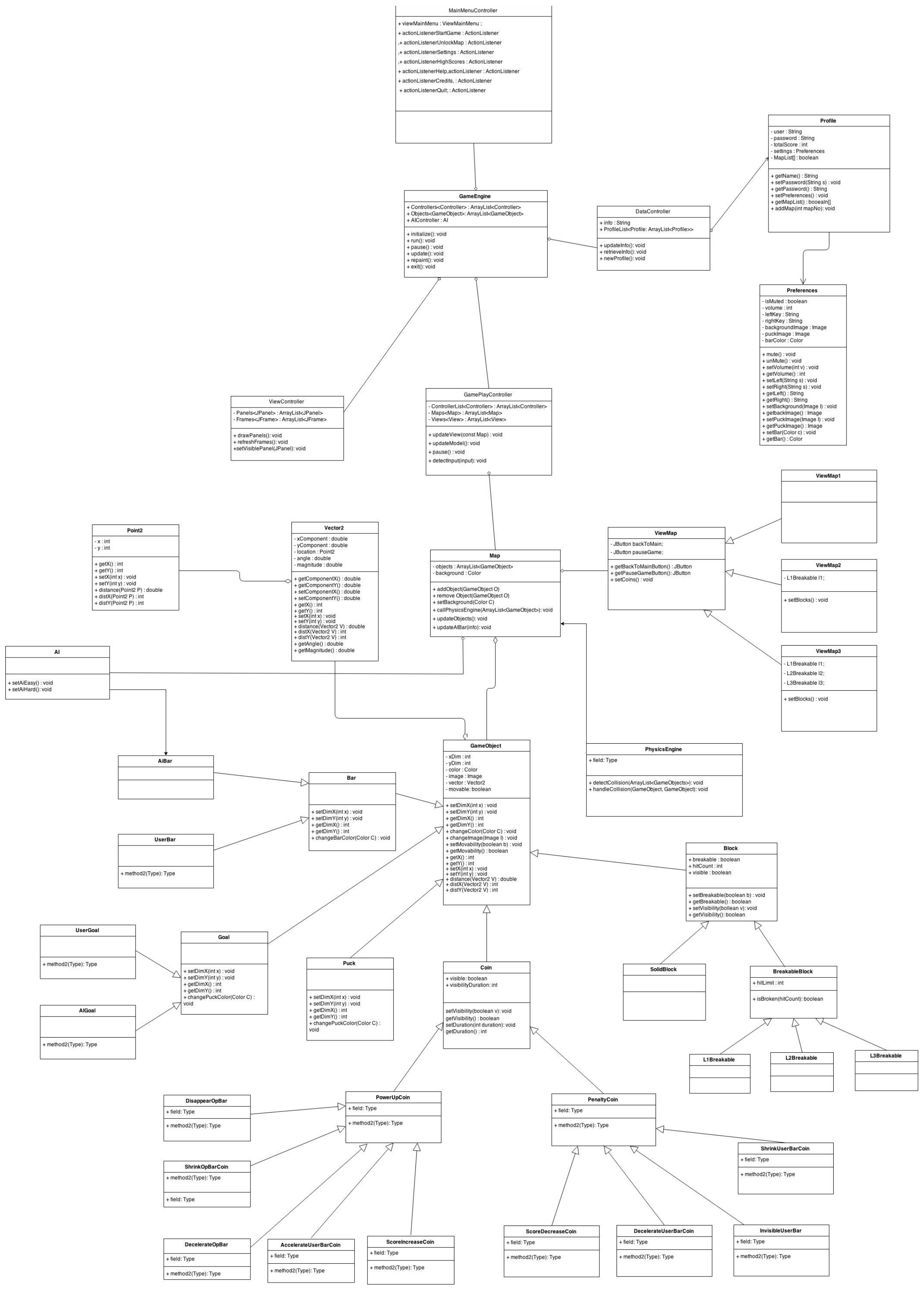
	5. System gives an error on the screen. User has coins which has earned from the games by collecting different types of coins. If user collects enough coin to purchase that particular map, user will be able to play in that map. However, if user does not have enough coin amount, system will popup an error message that indicates the reason of that particular error.
Entry Condition	**This use case extends AccessMarket use case. It is initiated if the user do not have enough points to unlock a map in AccessMarket use case, while purchasing a map.**
Exit Condition	**User clicks the back button.**
Quality Requirements	**The systems response time should be 1/10 seconds**

Scenario Name	AccessMarket
Participating Actor Instances	**Initiated by User**
Flow of Events	User clicks on Maps button while on the main menu screen to access the market.
	System displays the market screen. System checks the user profile and displays the maps that are not yet purchased and the ones that are already purchased with clear and blurred view, respectively. If user has purchased all of them, system does not let he/she to interact with any of the maps.
	System displays the maps without selecting none at the beginning.
	User clicks on one of the maps that appear with vivid colors, which indicate that map is available for purchase. User hits purchase button to buy the respective map.
	System checks the total score of the user and validates that user has enough points to buy the selected map. System updates user profile: adds the map to profile, deducts necessary amount of score and blurs the purchased map in the current market view.
	User presses back button and goes back to main menu screen.
Entry Condition	**This use case extends AccessMarket use case. It is initiated if the user do not have enough points to unlock a map in AccessMarket use case, while purchasing a map.**
Exit Condition	**User clicks the back button.**
Quality Requirements	**The systems response time should be 1/10 seconds**

Scenario Name	ExitToMainMenu
Participating Actor Instances	Initiated by User
Flow of Events	User is in an active game and game continues with its normal flow.
	User clicks on exit button on the play game screen.
	System displays a warning message, indicating that leaving the game in the middle of a match results in loss of all progress in this match.
	User presses “Exit anyways” button to leave the game.
	System deletes whole progress in the game and refreshes the objects in the map. System does not update the profile.
Entry Condition	**This use case extends AccessMarket use case. It is initiated if the user do not have enough points to unlock a map in AccessMarket use case, while purchasing a map.**
Exit Condition	**User clicks the back button.**
Quality Requirements	**The systems response time should be 1/10 seconds**

OBJECT MODEL

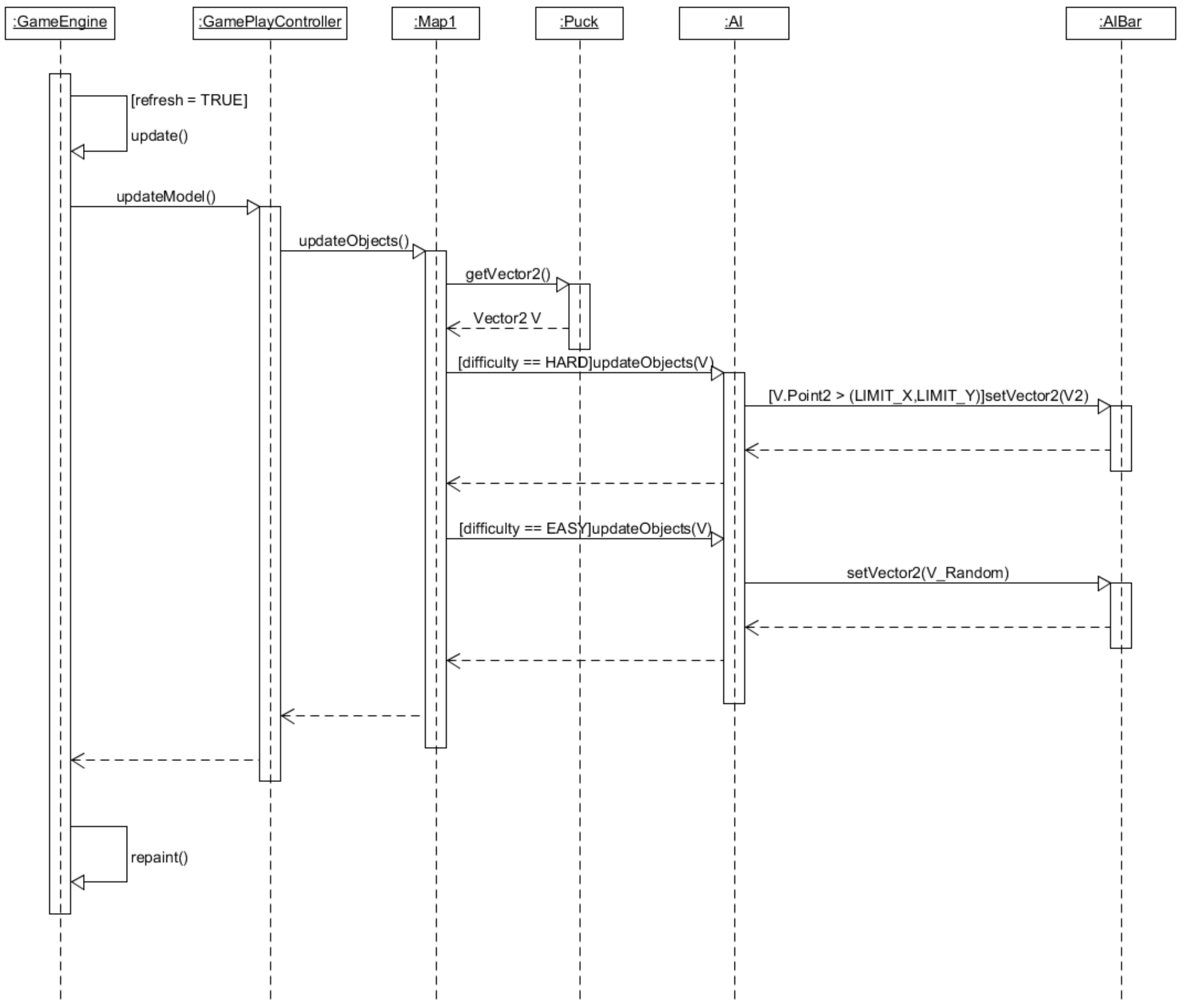
CLASS DIAGRAMS



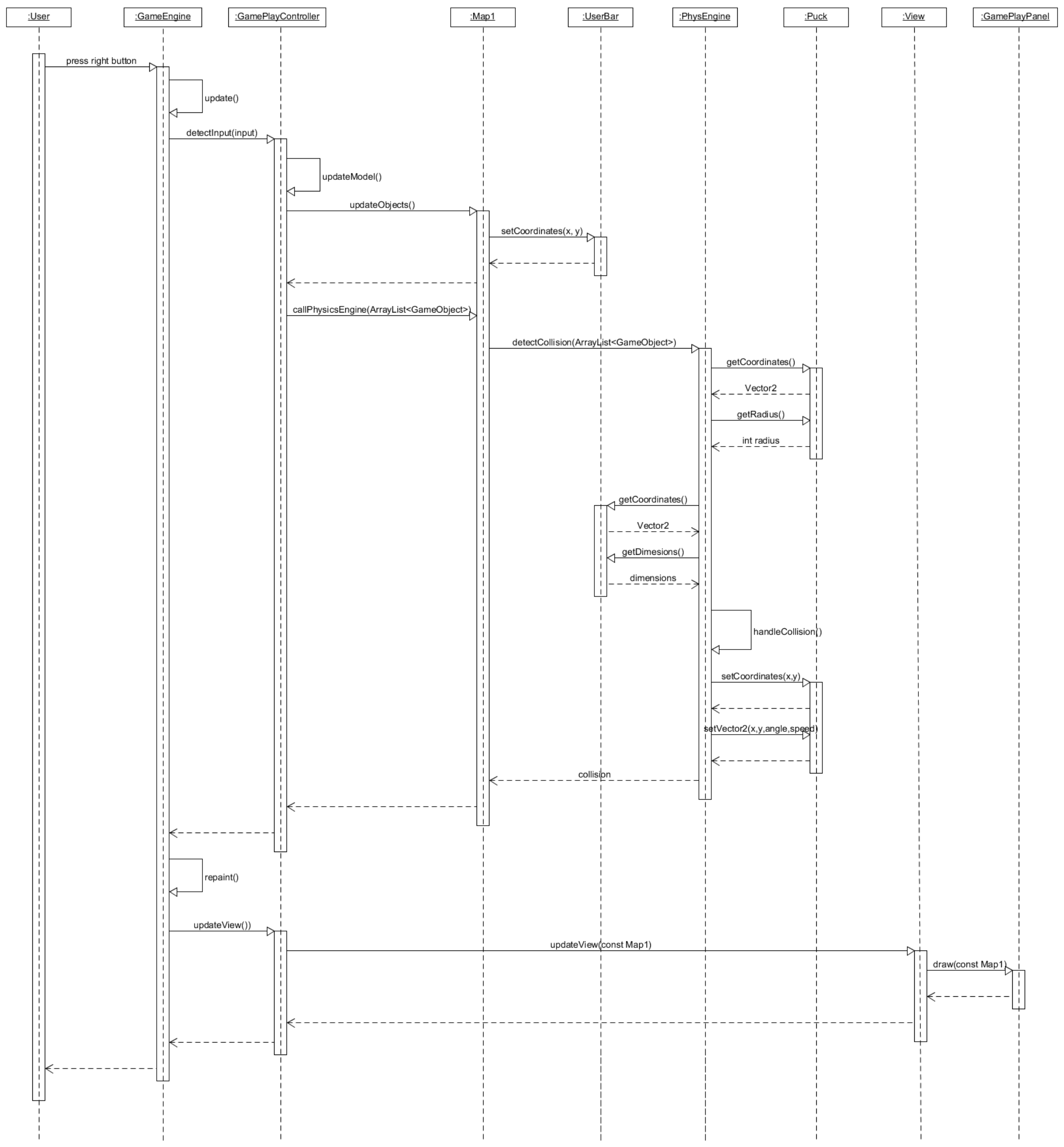
DYNAMIC MODEL

Scenario's related to each sequence diagram can be found under Scenario's title.

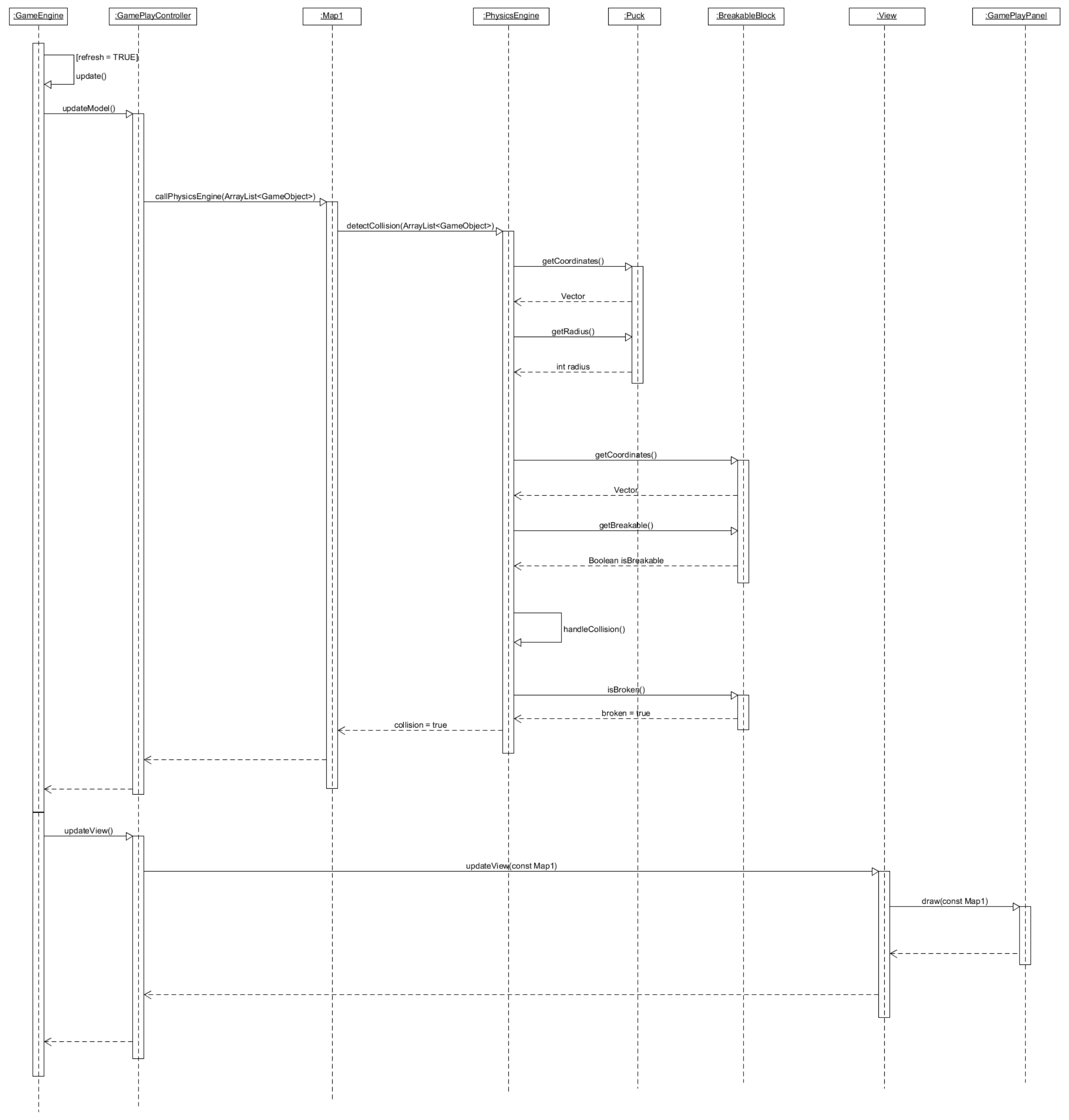
AIBarMovementUpdate



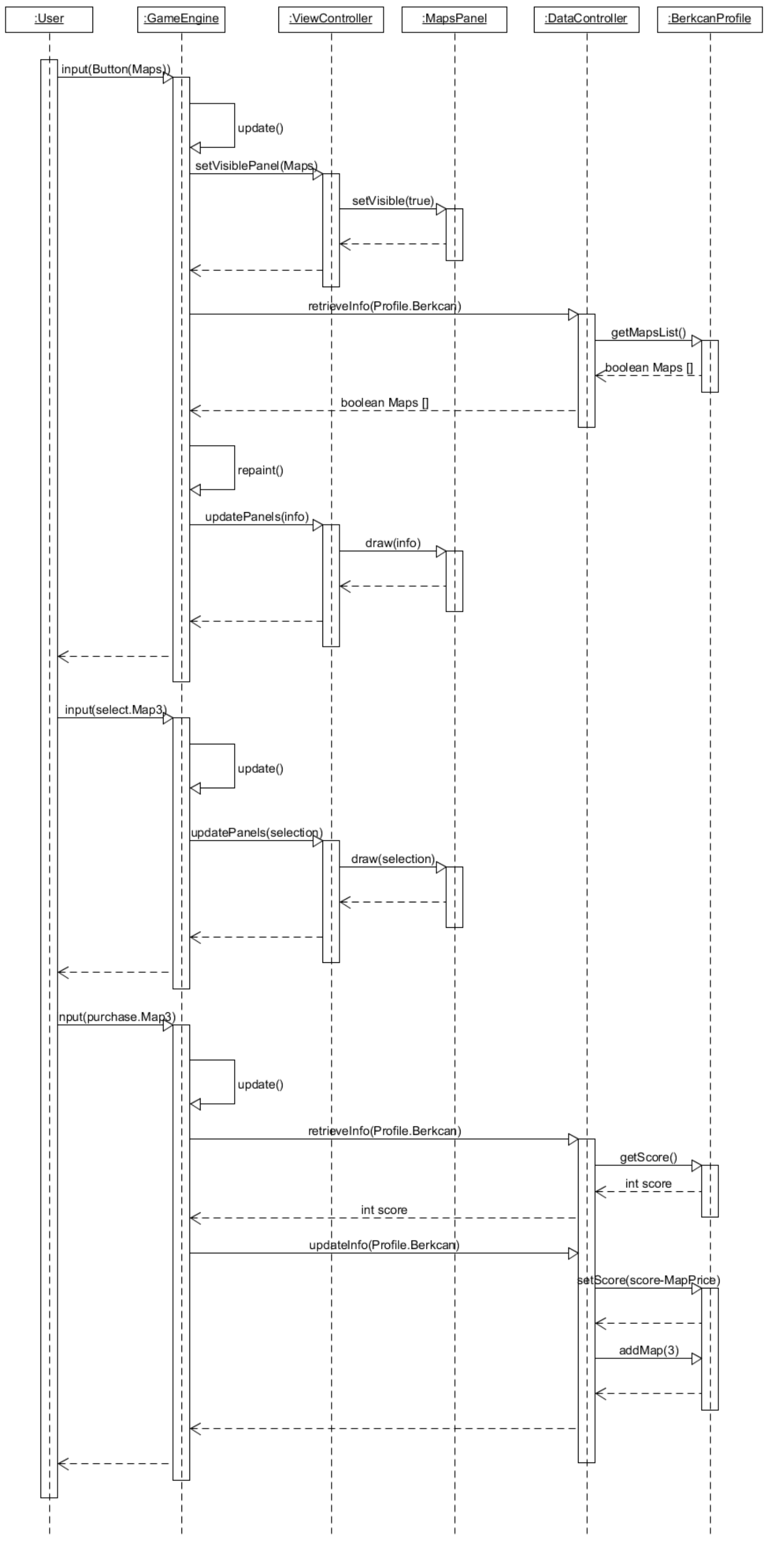
BallCollidePuck



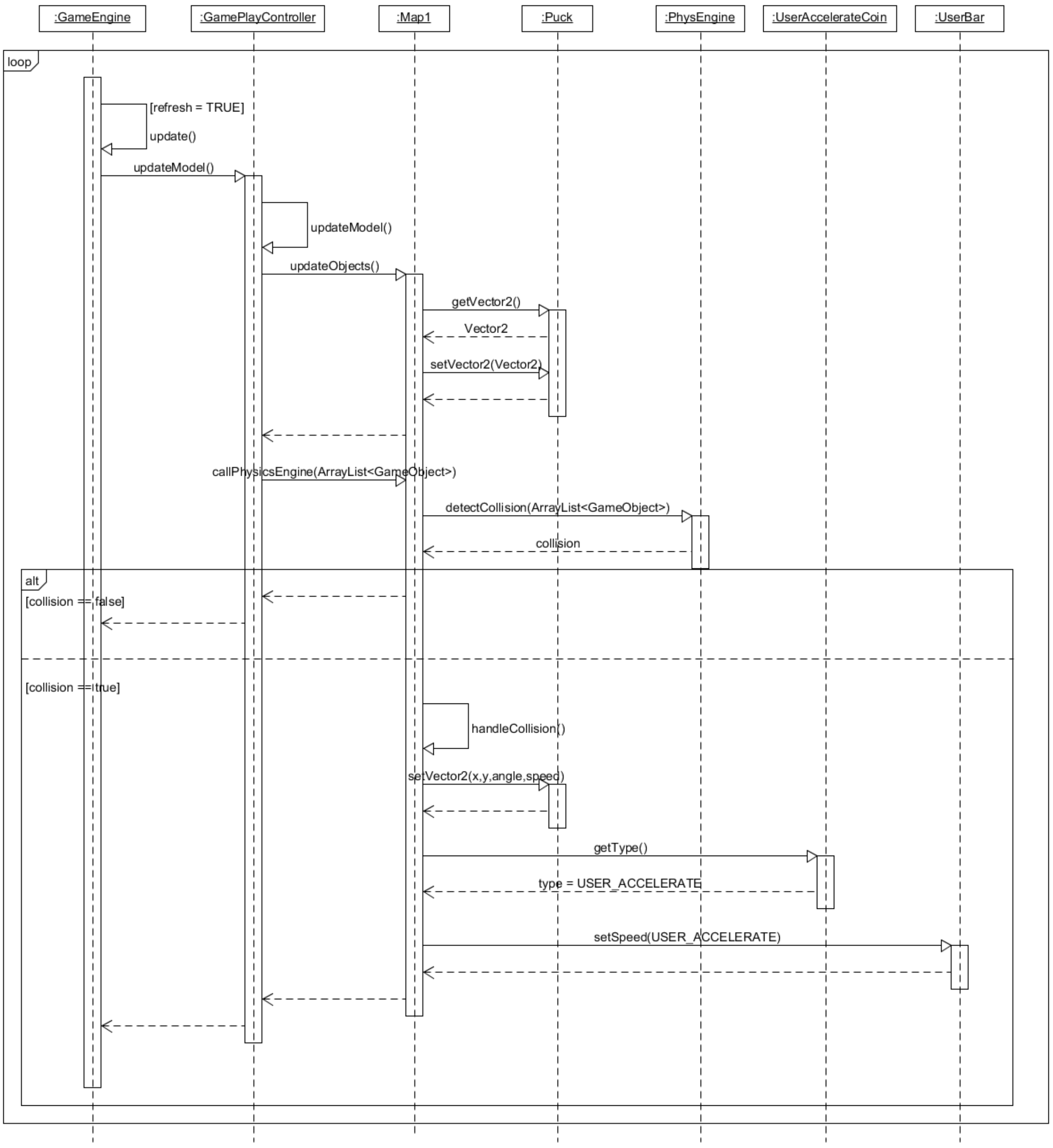
BreakBrick


:GameEngine
:GamePlayController
:Map1
:PhysicsEngine
:Puck
:BreakableBlock
:View
:GamePlayPanel
[refresh = TRUE]
update()
updateModel()
callPhysicsEngine(ArrayList<GameObject>)
detectCollision(ArrayList<GameObject>)
getCoordinates()
Vector
getRadius()
int radius
getCoordinates()
Vector
getBreakable()
Boolean isBreakable
handleCollision()
isBroken()
broken = true
collision = true
updateView()
updateView(const Map1)
draw(const Map1)

NewMapPurchase

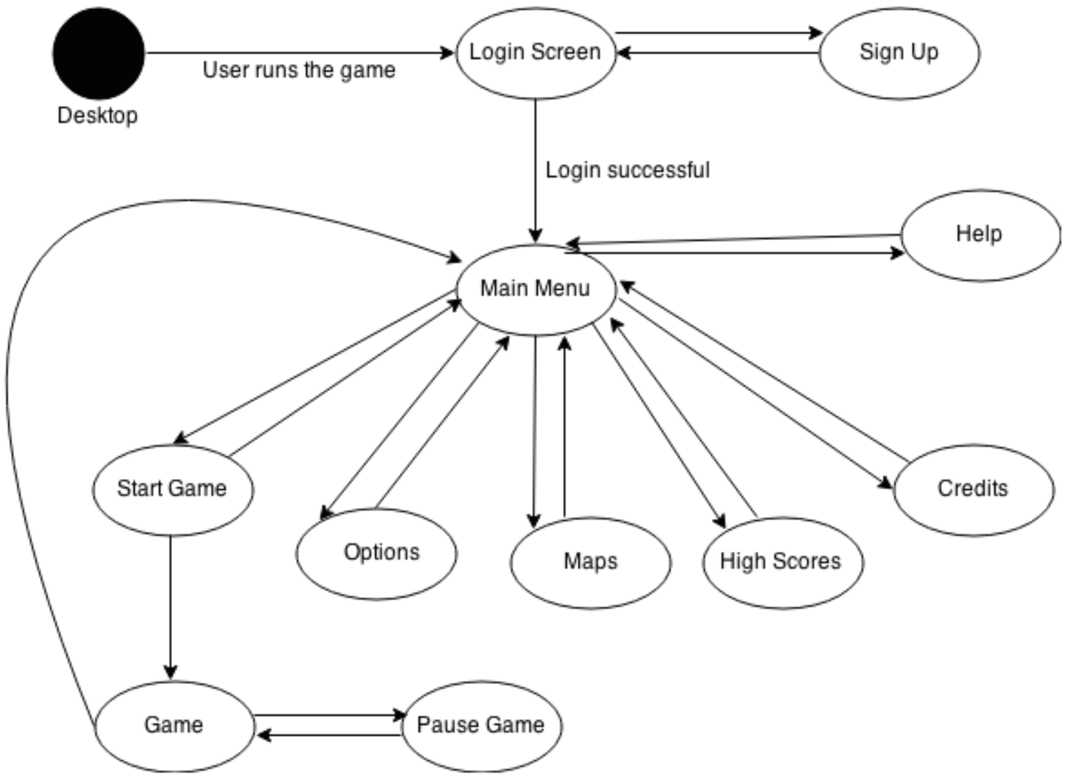


UserBarAccelerateCoinCollect

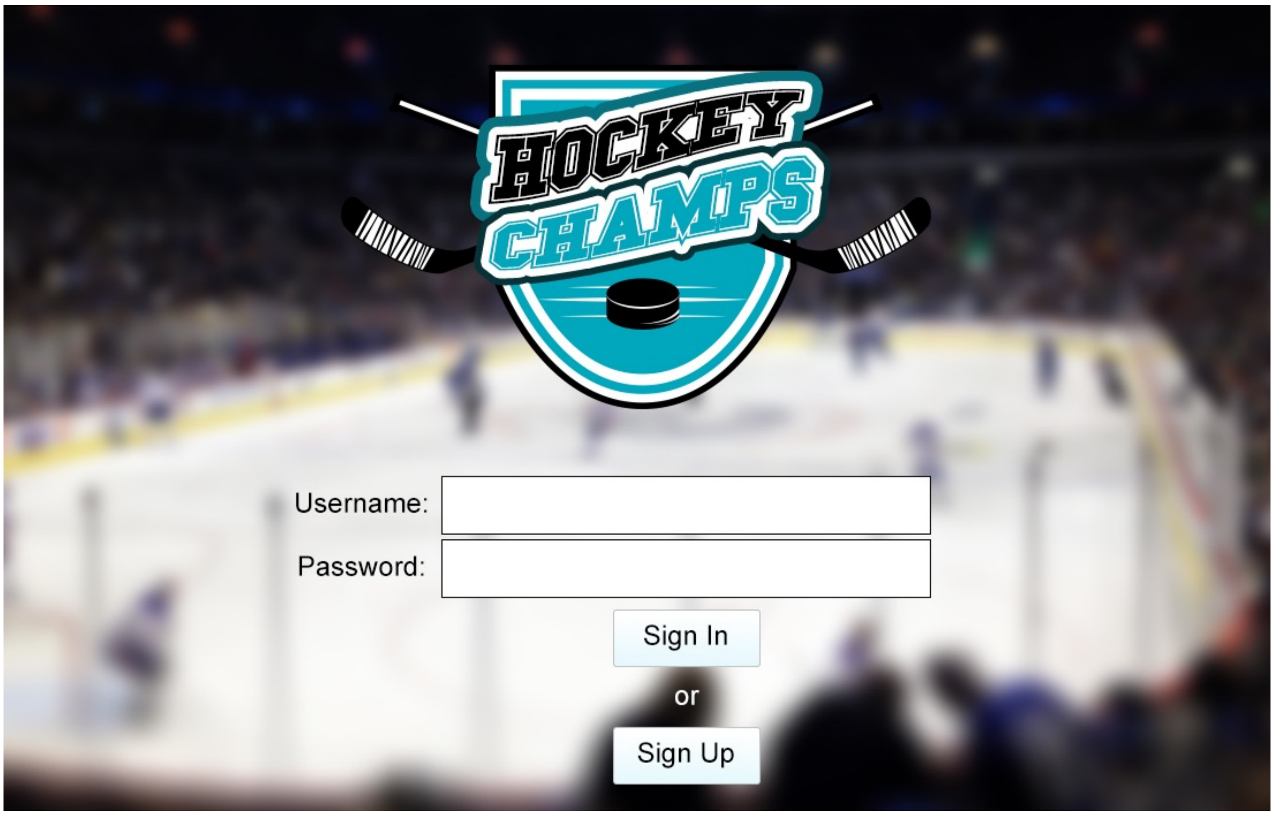


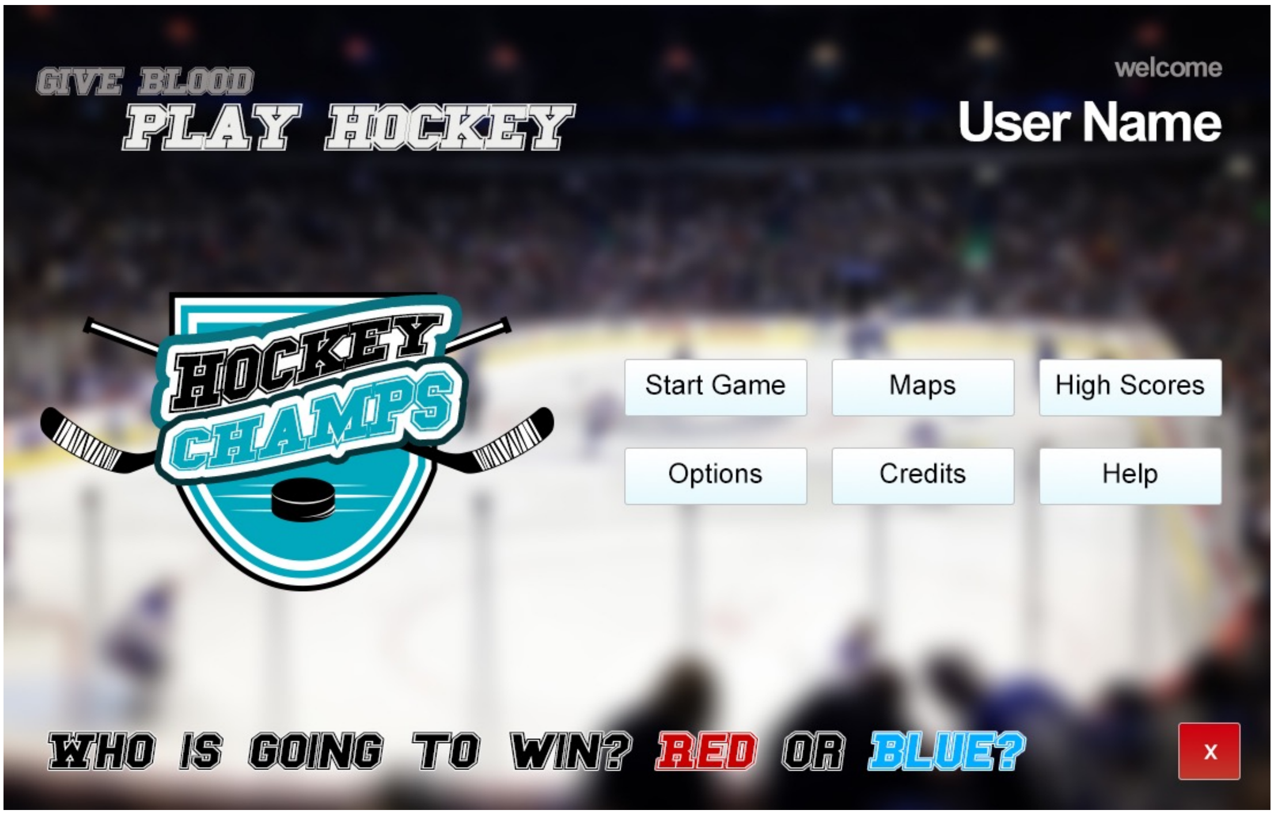
USER INTERFACE

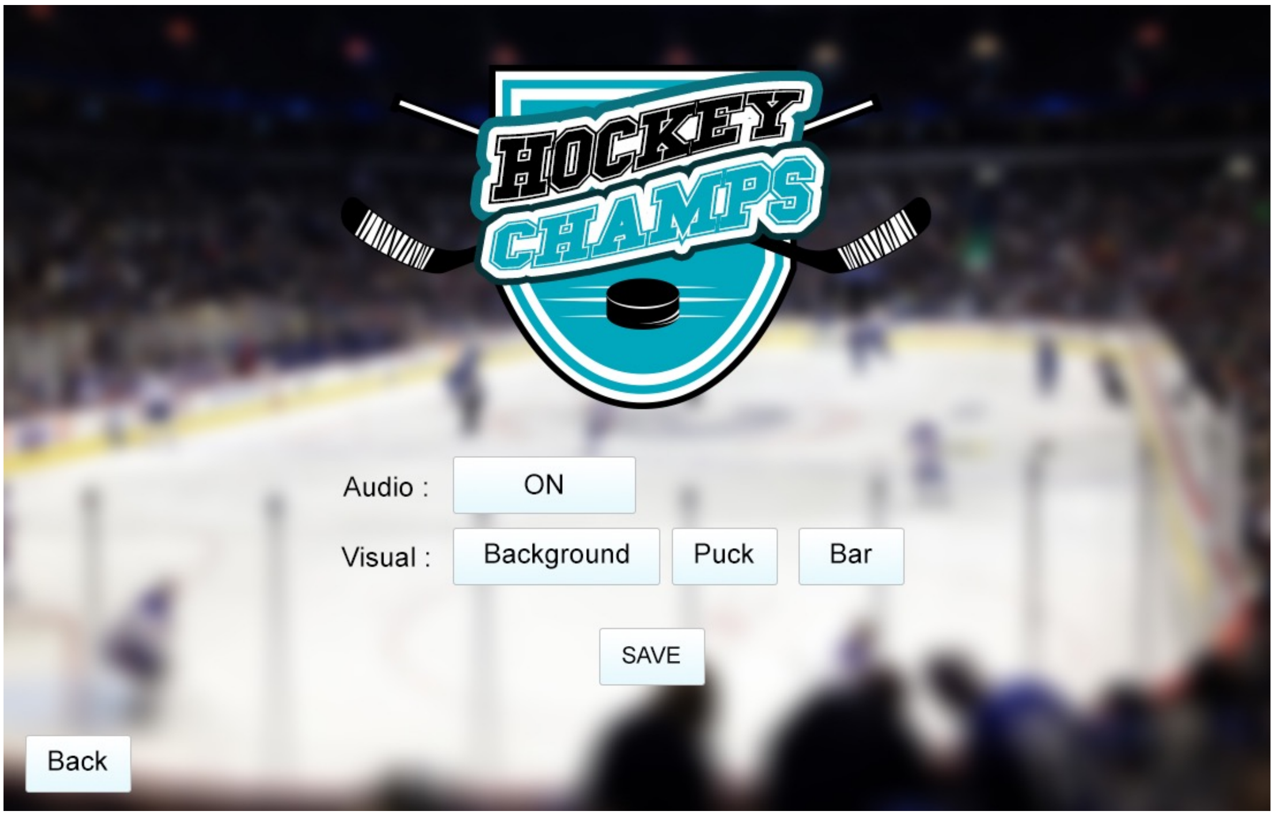
1. NAVIGATIONAL PATH

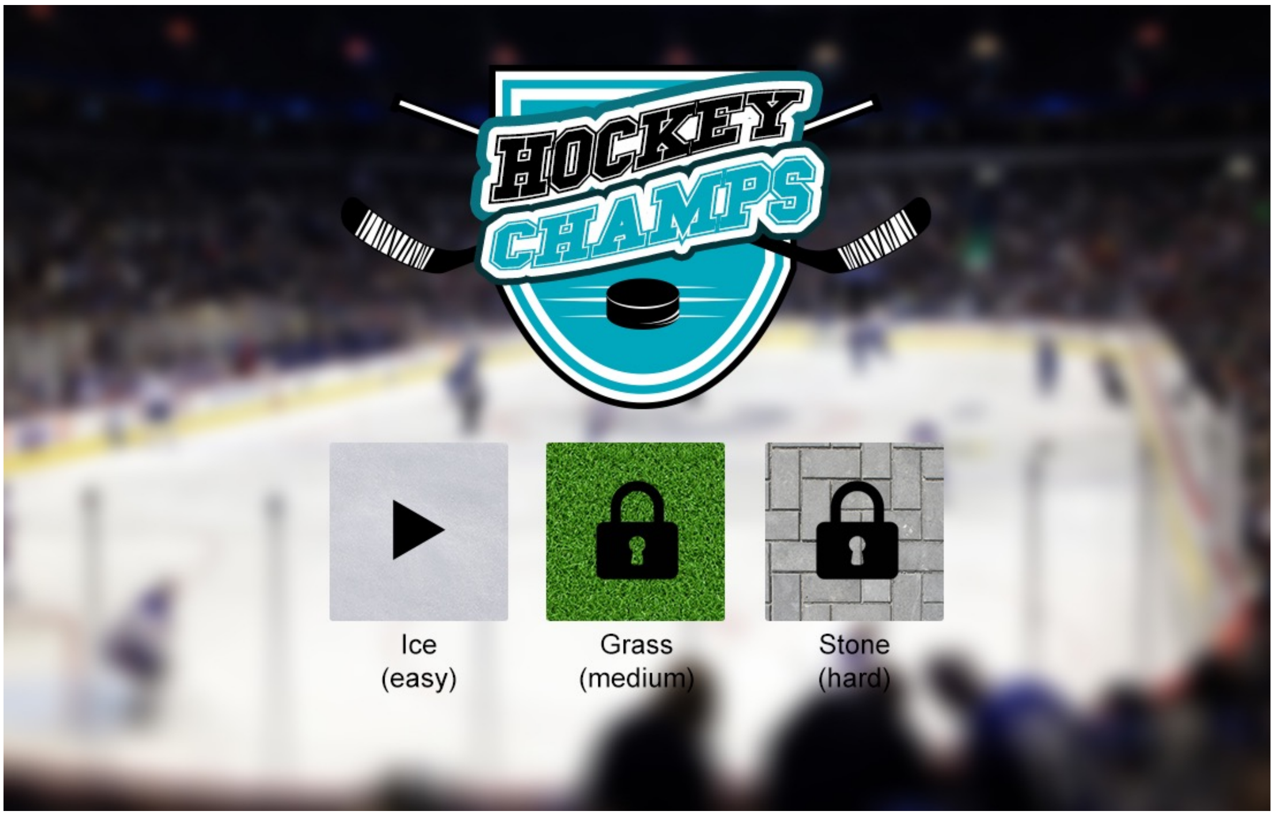


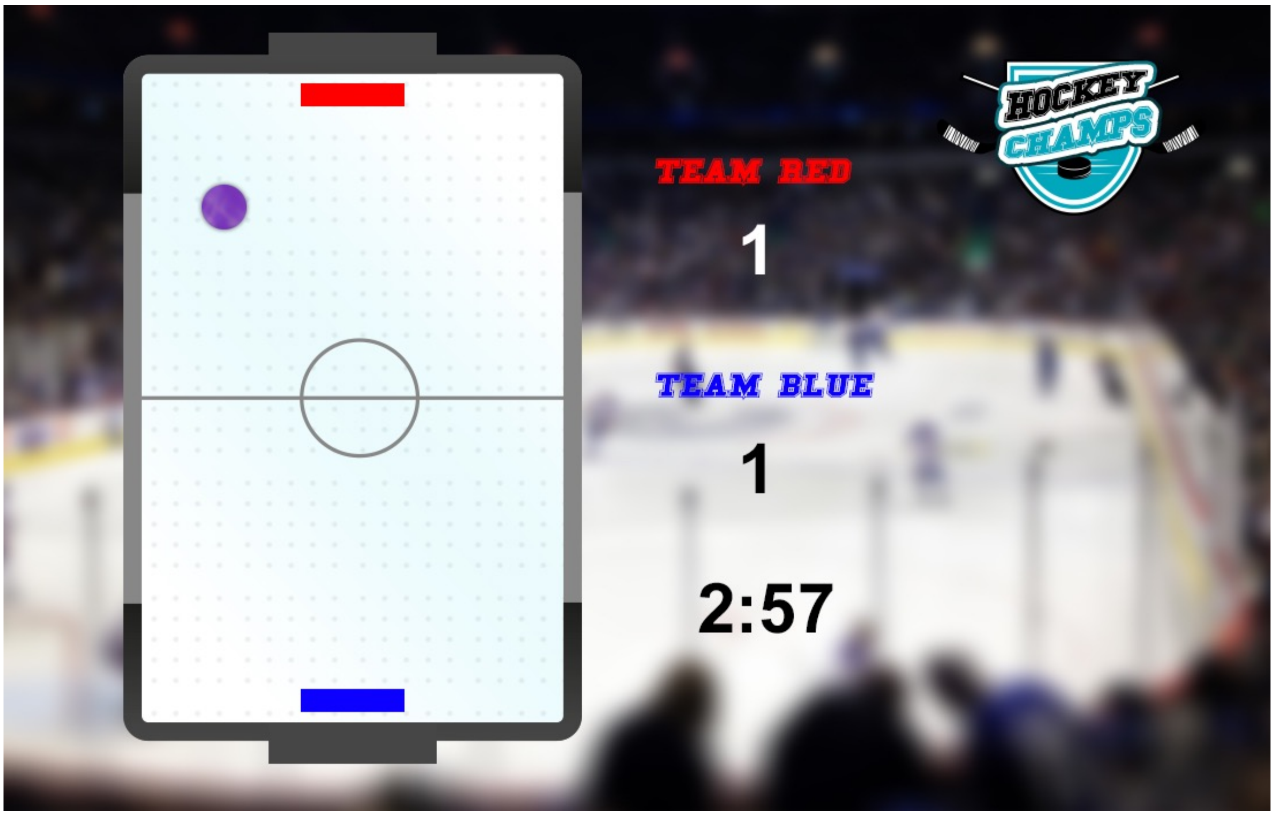
2. *SCREEN MOCKUPS*






HOCKEY CHAMPS
Audio :
ON
Visual :
Background
Puck
Bar
SAVE
Back


HOCKEY CHAMPS
Ice
(easy)
Grass
(medium)
Stone
(hard)


HOCKEY CHAMPS
TEAM RED
1
TEAM BLUE
1
2:57