המחלקה להנדסת תוכנה

שם הפרויקט: ביט מי משחק מציאות מדומה

Project Name: Beat Me VR Game

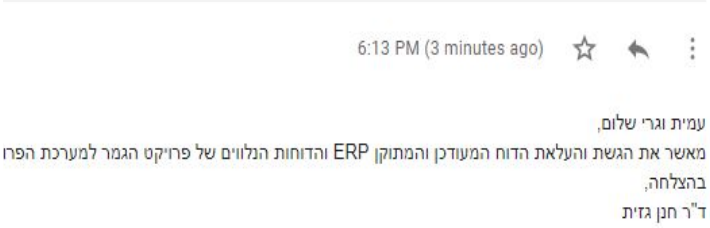
דוח ההנדסי:

שם סטודנט 1: עמית פלרמן

שם סטודנט 2: גבריאל דבח

שם המנחה: ד"ר אלחנן גזית

חתימת המנחה:



תאריך ההגשה: 23.01.2020

תוכן עניינים

3. תקציר

במשך שנים רבות מציאות מדומה הייתה חלום קשה למימוש בעולם פיתוח המשחקים. רק לפני עשור Oculus Rift הוציאו את אב הטיפוס הראשון שלהם והתחום היה לא מספיק בשל. כיום, עולם הגיימינג עובר שינוי משמעותי.

נתח השוק של משחקי מציאות מדומה הולך וגדל בעולם קונסולות המשחק המתקדמות, וגם רכישת ציוד הקצה הולך ונהייה נגיש יותר וזול לעומת ימים עברו.

משחק מציאות מדומה נותנת לשחקן את הרגשת ההתמעות IMMERSIVE להיות בתוך המשחק, הכי ראליסטית שניתן לקבל בהשוואה למשחקי מחשב או משחקי סלולר. בדיוק בשל סיבה זו בחרנו לפתח משחק מציאות מדומה.

תקציר זה מתאר את פיתוח משחק מציאות מדומה הנקרא "Beat Us".

המטרה העיקריות של הפרויקט אותו נפתח הינו משחק מציאות מדומה המבוסס בינה מלאכותית , AI, שתדע לנתח את המוזיקה ולהציג אובייקטים לפי הקצב.

יש לציין שפיתוח משחק כזה מורכב בשל האתגר ההנדסי שלו ומחדש מהמתחרים היום בשוק הקיימים בקונסולות שונות אותן סקרנו במהלך העבודה.

השחקן יצטרך להכות באובייקטים בהתאם למוטות אותו יאחז, להתחמק מאובייקטים מסוימים וכך יצבור נקודות כשהמטרה כמובן היא לצבור כמה שיותר ולעמוד בראש הטבלה.

המשחק מפותח ל PC שכן נבחר מבין החלופות המערכתיות בשל היותו מספק גרפיקה ברמה גבוהה וחווית משתמש בהתאם.

4. סקירה ספרותית וסקר שוק

אין שינויים או תוספות מהדוח התכנון.

5. תכן מפורט

פתרון שנבחר לתכן המערכת:

תיאור מקוצר של גרסת ALPHA:

- המשחק עובד בצורה תקינה ב Oculus rift S.
- ישנה מציאות מדומה ב3D.
- ישנם 4 אובייקטים שונים (כחול,אדום ושני מסיחים) המגיעים לפי קצב המוזיקה.
- ניקוד השחקן מתוגמל לפי ההצלחה שלו להכות באובייקטים והתחמקות ממסיחים.
- ניקוד השחקן מוצג בזמן אמת בשול השמאלי העליון של המסך.
- ישנן שלוש רמות קושי:
  - קל: רמת קושי זו מותאמת לשחקנים מתחילים, במהירות איטית, אובייקטים מגיעים בבודדים ואין מסיחים.
  - בינוני: ברמת קושי זו מהירות האובייקטים גדלה ונוספו מסיחים.
  - קשה: ברמת קושי זו המהירות הכי גדולה של האובייקטים, יש יותר מסיחים מאשר ברמה הבינונית וגם לפעמים מגיעים 2 אובייקטים במקביל.

מה נותר:

- לשפר עיצוב המשחק והאובייקטים.
- לחבר בסיס נתונים למשחק ולפיו לעדכן את טבלת התוצאות.
- שיפור רמות הקושי.
- המרת סקריפט האלגוריתם מSCALA ל#C והטמעתו בקוד ובהתאם לביצועים אולי לתת לו לעבוד "אונליין" כאשר המשתמש מעלה שיר (בהתאם לבדיקות ביצועים).
- בדיקות מקיפות על כל חלקי הקוד והמשחק, אינטגרציה וכו'.

אלגוריתם:

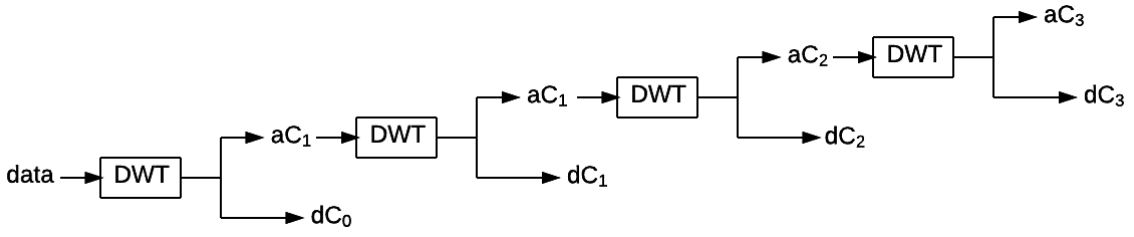
האלגוריתם מיושם במערכת הינו אלגוריתם לגילוי טמפו של שיר BPM - beats per minute.

בשלב זה האלגוריתם כתוב בשפת Scala ועובד רק עם קבצי wav. אך לקראת הבטא אנחנו מתכוונים לנסות להמיר את כולו או חלקו לשפת #C ולנסות לתמוך בסוגי קבצים נוספים, כל זאת כמובן בהתאם לביצועים ולספריות התומכות ב-#C אם וכאשר.

לגבי האלגוריתם עצמו:

האלגוריתם שלנו מבוסס על התמרת DWT. התמרה של קובץ אודיו ממרחב הזמן למרחב התדר.

האלגוריתם מחלק את קובץ לפריימים. גדלי הפריימים מייצגים 1-10 שניות מהקובץ המקורי. כל פריים מעובד על ידי פילטר של DWT, ואז מחולק לארבע רצועות משנה, שהם ארבעה שלבים מקוננים של DWT.



עבור כל dC מחשבים מעטפה על ידי שלוש פעולות: ערך מוחלט, דאון סמפלינג, ונירמול(הורדת הממוצע).

כל המעטפות בפריים מסוכמות ועוברות אוטוקורלציה. כל שיא באוטוקורלציה מתכתב עם שיא בפריים המקורי.

ברגע שכל הפריימים מחושבים, הערך המוחזר הוא החציון, BPM ביטים לדקה.

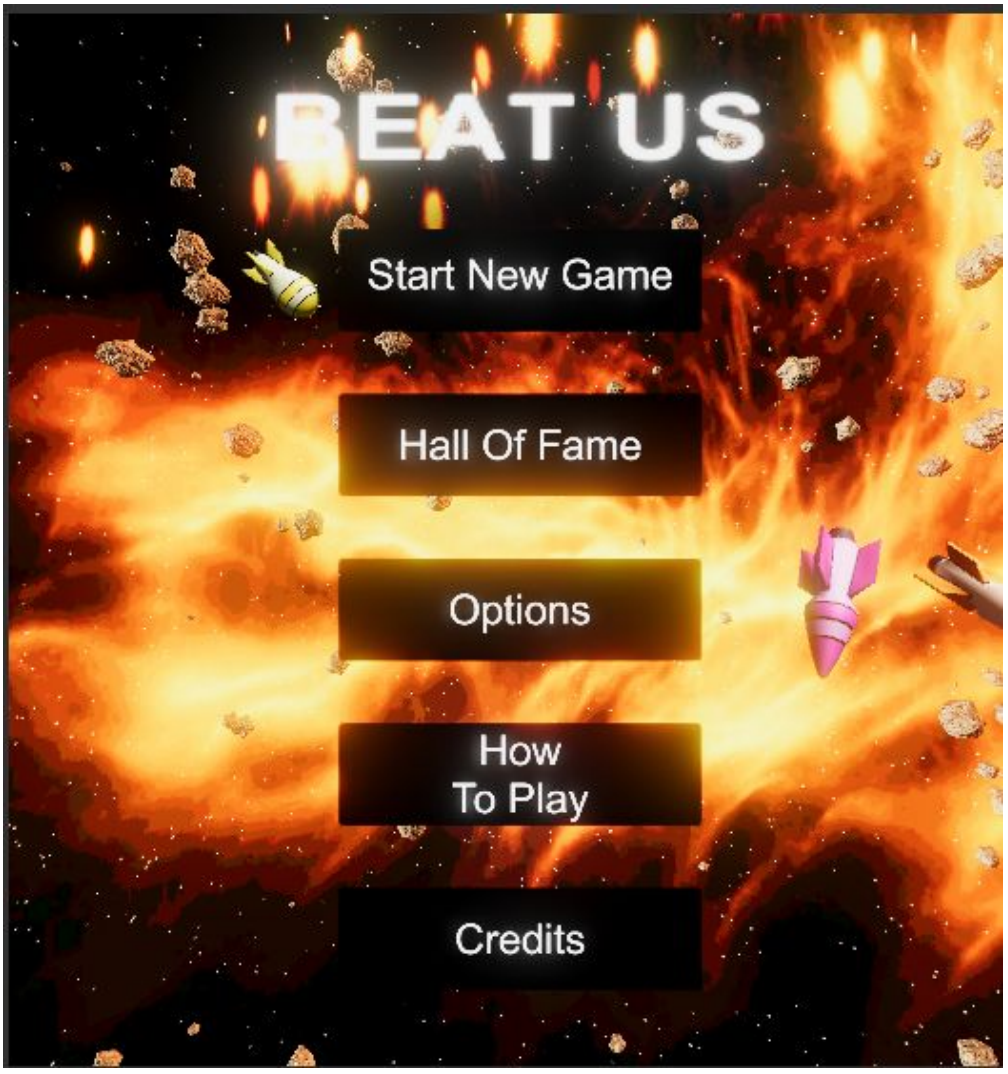
תוצאות:

את תוצאת האלגוריתם אנחנו מיישמים במשחק בתוך ה-Spawner שמחשב את המרחק המתאים בין כל שני אובייקטים באמצעות ה-BPM שמקבל מהאלגוריתם.

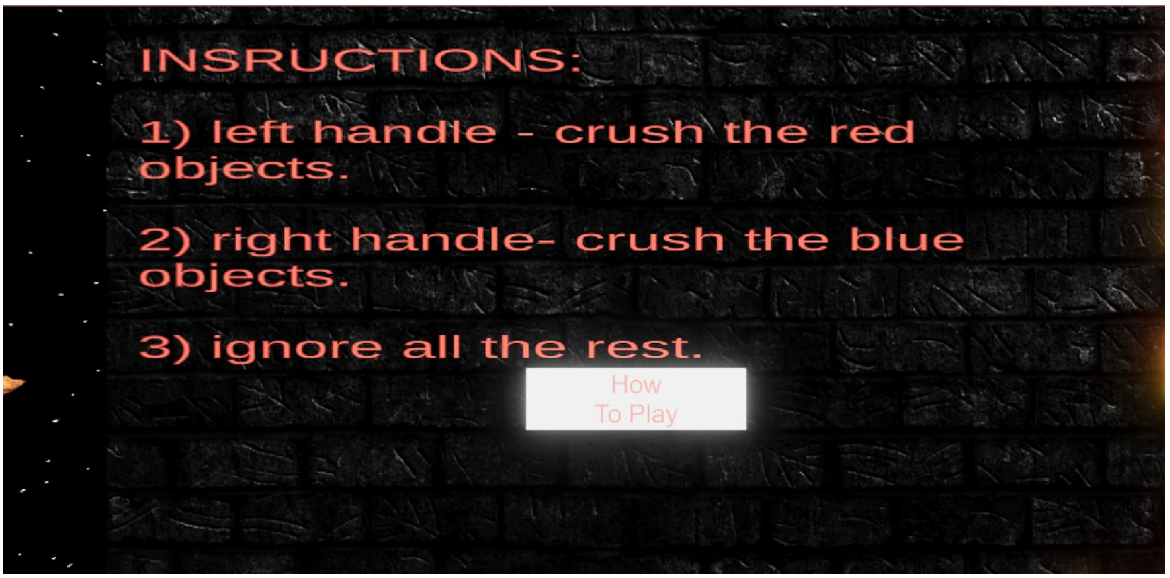
הנתון שמקבל הינו נכון ליחידות של דקה ואילו ה-Spawner משגר אובייקטים רבים בזמן הזה ולכן הוא עושה המרה על ידי חילוק של 60 שניות ב-BPM וגם מכפיל ב-2 על מנת למצוא את המרחק בין שני אובייקטים שישוגרו.

התפריט הראשי:

כאן המשתמש בוחר האם להתחיל משחק חדש/ לצפות בטבלת התוצאות/ לקרוא כיצד לשחק/ לבחור שיר ודרגת קושי/ לצפות בקרדיט.



הסבר איך לשחק:

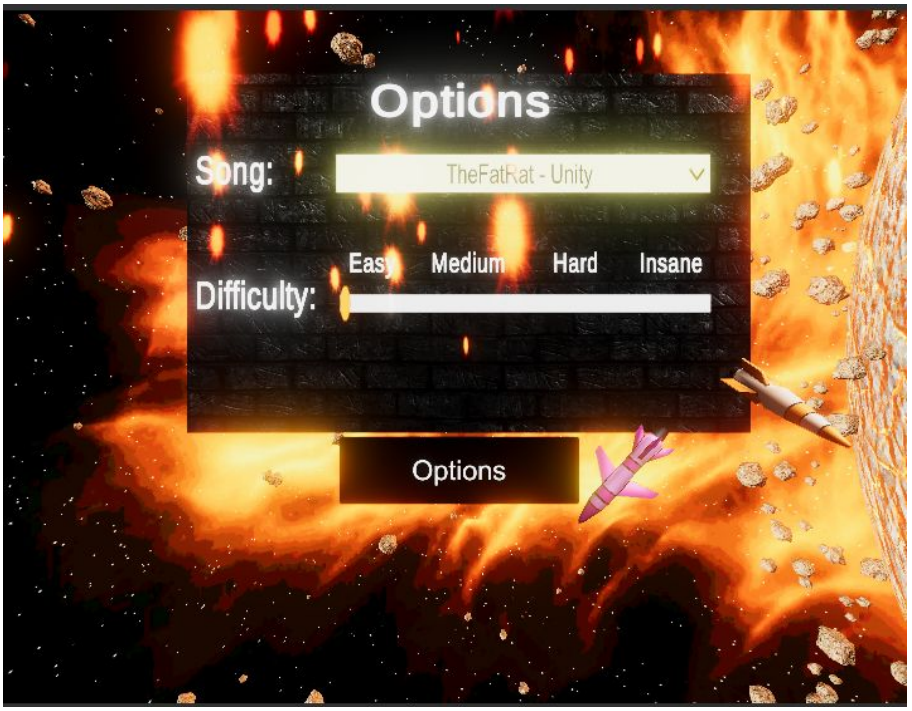


מסך טבלת התוצאות:

התוצאות נשמרות ונטענות בJSON.

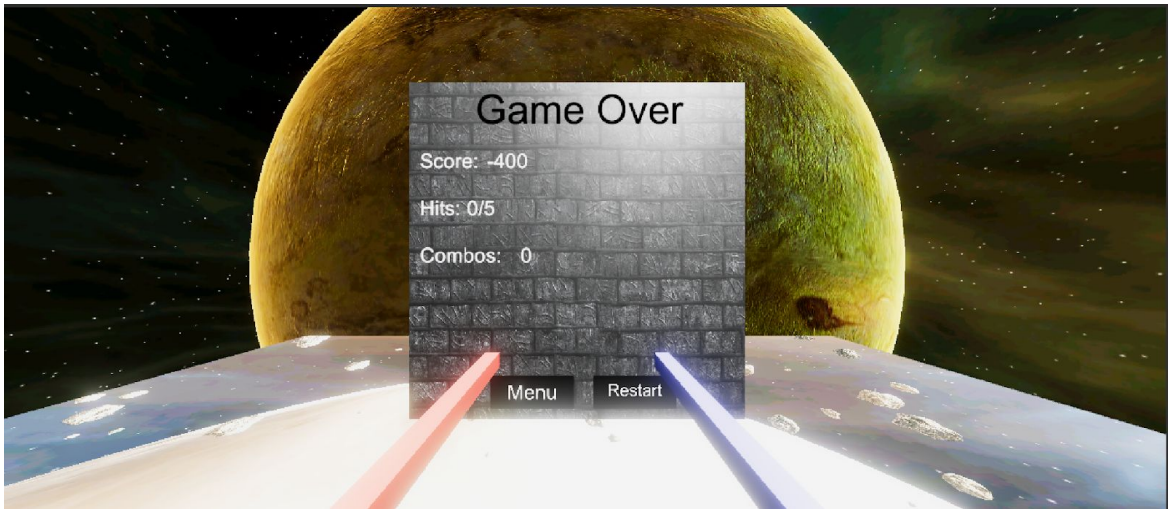
לחיצה על אפשרויות במסך הראשי תוביל את המשתמש לבחור שיר ורמת קושי:

Main->Options:



צילום מהלך המשחק (כנצפה בתוך משקפי המציאות המדומה):

צילום סיום המשחק והצגת התוצאה לשחקן:



6. תוכנית עבודה לביצוע הפרויקט

<u>תוכנית עבודה לאחר הגשת דוח הנדסי עד לסיום הפרויקט:</u>

מסירת הפרוייקט + דוח סופי	**89 days**	**Thu 23/01/20**
עדכון מסמכים קודמים	2 days	Thu 23/01/20
אפיון המערכת	1 day	Fri 24/01/20
כתיבת פירוט מוצר	3 days	Sat 25/01/20
כתיבת תכנון הפרוייקט	8 days	Tue 28/01/20
בדיקות והערכות	8 days	Thu 06/02/20
פיתוח קוד	**66 days**	**Tue 18/02/20**
פיתוח המשחק	45 days	Tue 18/02/20
אינטגרציה בין חלקי הקוד	12 days	Tue 21/04/20
בדיקות המערכת	8 days	Thu 07/05/20
הגשת דוח סופי	1 day	**Tue 26/05/20**

7. ריכוז שינויים מאז דוח התכנון:

לאור התנסות ובדיקות שערכנו, החלטנו לשקול האם לוותר על חיבור לבסיס נתונים לצורך שמירת שיאים ולשמור אותם מקומית בעזרת ב-JSON.

בשל כך שאנחנו שומרים בסך הכל 10 שיאים שכוללים שני תאים (שם ותוצאה), זו פעולה מהירה וקלה לעשות ב-JSON, שלא מצריכה שימוש בבסיס נתונים שעלול להכביד על המערכת.

8. ניהול סיכונים:

8.1 הסיכונים המרכזיים הצפויים ודרכי התמודדות

8.1.1 סיכון 1:

האטה והכבדה על זמן התגובה של המשחק משום האלגוריתם המורכב לפענוח קצב המוזיקה.

דרכי התמודדות:

8.1.1.1 לחפש כיצד אפשר לייעל את האלגוריתם ככל האפשר מבלי לפגוע באיכות הביצוע.

8.1.1.2 פיתוח אלגוריתם שניגש לבעיה בצורה שונה.

8.1.2 סיכון 2:

תקינות האוקולוס ריפט שברשותנו. ישנו מכשיר אחד במעבדה, מכיוון שכל יכולות המשחק במציאות מדומה תלויות בתקינות המכשיר ברשותנו.

דרכי התמודדות:

8.1.2.1 לשמור את המכשיר בצורה המיטבית.

8.1.2.2 להתכונן להציג את המשחק גם בצורה שאינה מציאות מדומה.

8.1.2.3 לדאוג למכשיר חלופי.

9. רשימת מקורות.

Levels of Sound: On the Principles of Interactivity in Music Video Games.

M Pichlmair, F Kayali - DiGRA Conference, 2007

Pitch extraction and fundamental frequency: History and current techniques

D Gerhard

Department of Computer Science, University of Regina

- Tzanetakis, George & Essl, Georg & Cook, Perry. (2001). Audio Analysis using the Discrete Wavelet Transform. Proceedings of the Conference in Acoustics and Music Theory Applications. 318-323.

10. נספחים

10.1 מסמך SRD

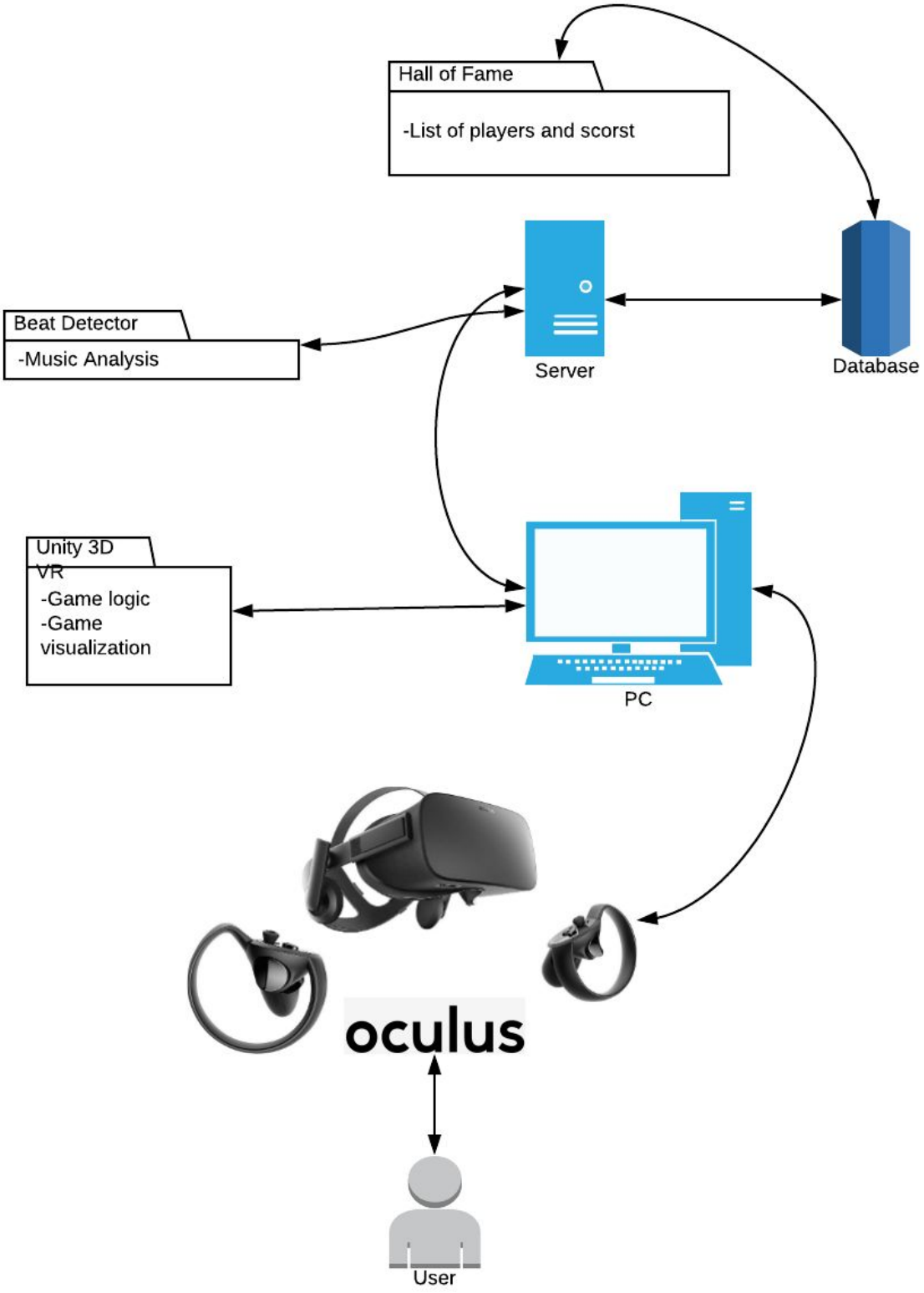
Software Requirements Document

1. **Introduction**

   Our game is a VR game, which requires a PC and Oculus VR system, the goal of the game is to hit objects with two swords (controlled by Oculus handles). The objects thrown by the tempo of the music. Each hit on the objects rewards with extra points.

2. **Model description**

   Our model is a VR game using Oculus headset and handles. The backbone of the production of the game is with Unity game engine and development IDE, and our own music analysis module. The architecture is presented in the diagram below:


Hall of Fame
-List of players and scorst
Server
Database
Beat Detector
-Music Analysis
Unity 3D
VR
-Game logic
-Game
visualization
PC
oculus
User

the reasons for this architecture are obvious:

1. Unity is a free and very popular game engine and development especially for beginners and small teams, everything is “out of the box” solution for development.
2. Oculus is available in our labs, and has very good connectivity with Unity.

3. Specific Requirements

3.1 Functional Requirements

Actors & Goals

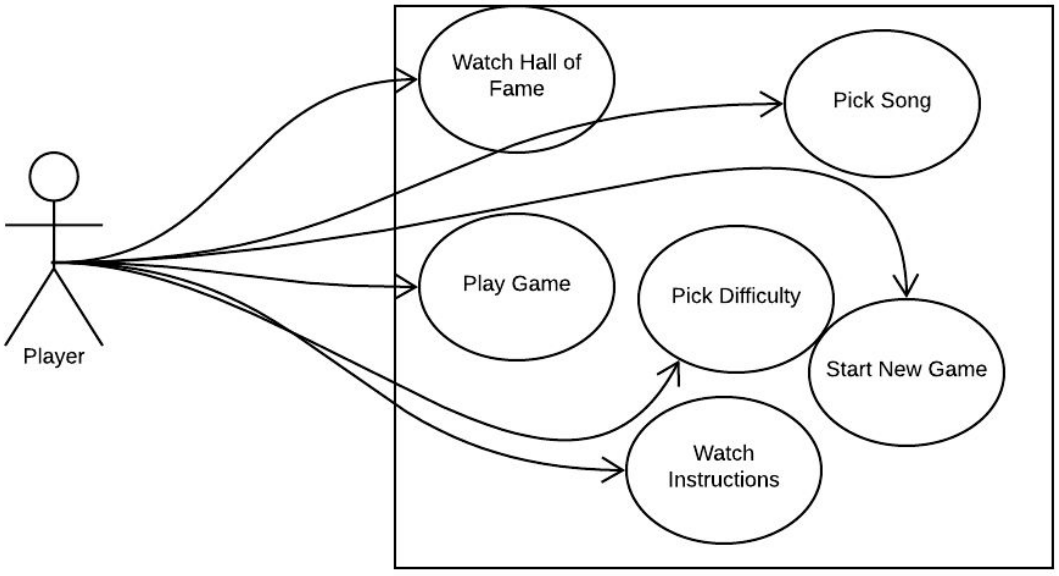
Actor name: Player

Type: primary

Goals:

- Play games
- Watch other players scores

Use Case Diagram



Use Cases Details

Start New Game

Main Actor: Player

Goal: the Player wishes to start a new game, System generate new game for Player.

Basic flow:

● Player enters game ● Player wish to start a new game	
	● System Generate a new game for Player in the level she chose

Play Game

Actor: Player

Goal: Player wants to play the game

Basic Flow:

● Player is in the game screen	
	● Game start throwing objects at the player
● Player tries to hit objects and win points ● Player tries to avoid hitting obstacles	
	● Game awards Player with points for every correct hit ● Game ends ● System show player his records
● Player leaves game	

Alternate Flow:

● Player is in the game screen	
	● Game start throwing objects at the player
● Player tries to hit objects and win points ● Player tries to avoid hitting obstacles	
	● Game awards Player with points for every correct hit ● Game ends ● System show player his records ● System notice that Player hit a record score ● System ask Player's nickname
● Player gives a nickname	
	● System saves the record with the nickname given by Player
● Player leaves game	

<u>Watch Hall of Fame</u>

Actor: Player

Goal: Player wants to see her and other players records

Basic Flow:

● Player wishes to watch top game records	
	● System show Player top records of the game
● Player watches the top records	

Watch Game Instructions

Actor: Player

Goal: the player wants to know how to play the game

Basic flow:

● Player wants to know how to play the game	
	● System Show Player the game instructions
● Player learn the game rules	

Pick Song

Actor: Player

Goal: the player set a song for the game

Basic flow:

	● System ask Player to pick a song
● Player picks song	
	● System saves the song preference

Pick Difficulty

Actor: Player

Goal: the playerset the level of difficulty for the game

Basic flow:

	● System ask Player to pick
● Player picks level of difficulty	
	● System saves the level of difficulty

3.2 Non-Functional Requirements

3.2.1 Performance requirements

- Less than 150 ms average reaction time for the player action

3.2.2 Resource requirements

- Windows 10 OS
- Oculus Rift VR System

3.2.3 Security requirements

The whole game is installed locally and in addition there is no sensitive information stored in this application, therefore there is no security requirements.

Software Design Description

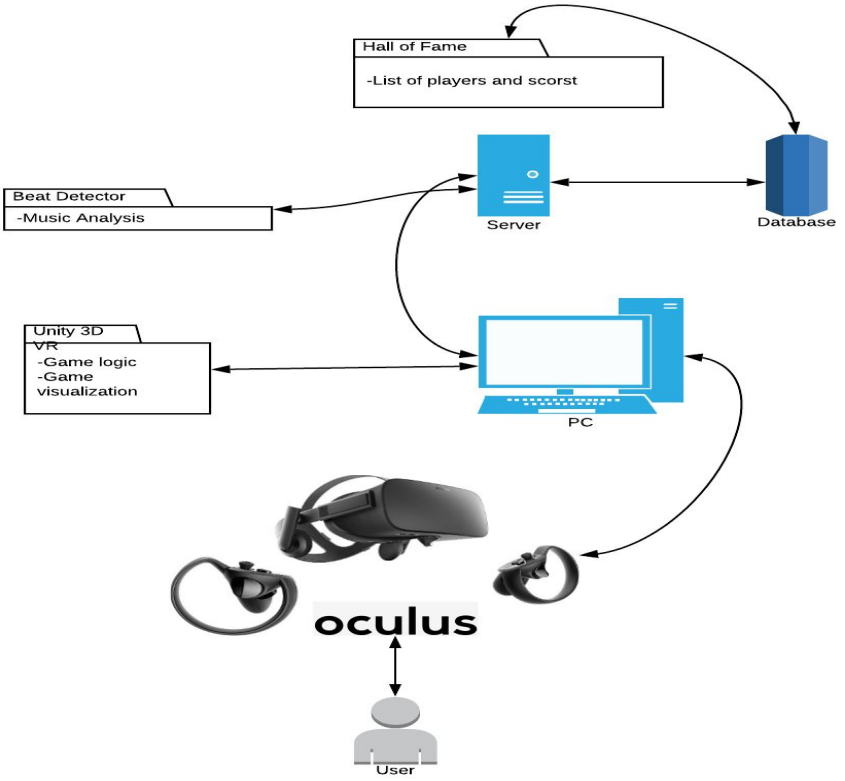
1. Introduction

Our game is a VR game, which requires a PC and Oculus VR system, the goal of the game is to hit objects with two swords (controlled by Oculus handles). The objects thrown by the tempo of the music. Each hit on the objects rewards with extra points.

2. System Architectural Design

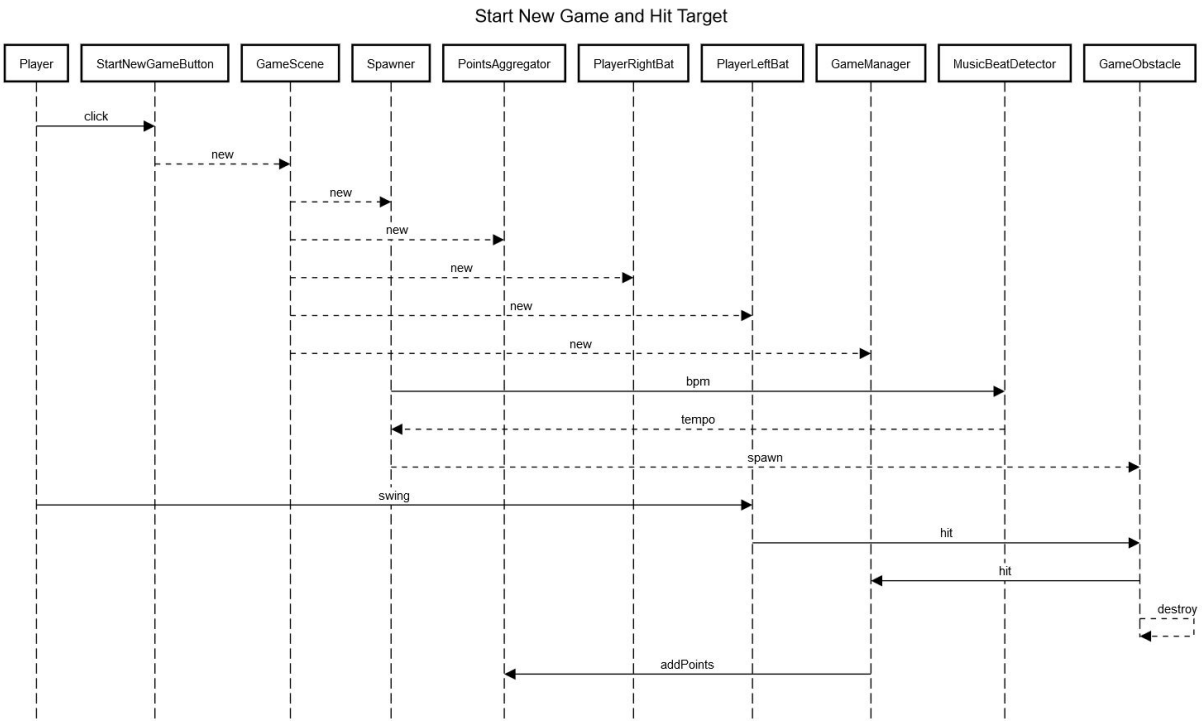
2.1 Chosen System Architecture

Our model is a VR game using Oculus headset and handles. The backbone of the production of the game is with Unity game engine and development IDE, and our own music analysis module. The architecture is presented in the diagram below:



2.2 System Interface Description

Start New Game



3. Detailed Description of Main Components

3.1 Game Scene

3.1.1 Type: The main screen of the game, 3D environment

3.1.2 Purpose: Where the game is played

3.1.3 Functional Inputs & Outputs: Present to the player the game objects in 3D and music. The Player moves and hit with his handles.

3.1.4 Interfaces: Seen on the Oculus headset goggles, and react to the player movement and controls.

3.1.5 Data: there is data that the Unity modules manages.

3.2 Music Beat Detector

3.2.1 Type: A class that an object of this class gets the audio file of the game.

3.2.2 Purpose: gets the audio file of the music and analyze its tempo for the timed objects that thrown at the player direction.

3.2.3 Functional Inputs & Outputs: audio file as input, tempo as outputs.

3.2.4 Interfaces: Through a script the game scene ask for the tempo and sends the audio file.

3.2.5 Data: The audio file is translated to an array, and the tempo that returns is an array of frames of tempo (every few seconds of the song).

3.3 Spawner

3.3.1 Type: An object on the scene(not seen).

3.3.2 Purpose: Creates objects(cubes and obstacles) randomly, and throw them towards the player position.

3.3.3 Functional Inputs & Outputs: Input: the tempo of the music of the game level. Output: gameobject - cubes and obstacles.

3.3.4 Interfaces: Not visible to the player.

3.3.5 Data: Unity handles behavior.

3.4 LeftBat and RightBat

3.4.1 Type: Two gameobjects 3D bats.

3.4.2 Purpose: Hit the objects thrown at the player.

3.4.3 Functional Inputs & Outputs: The player swings game handles supplied by Oculus. The game logic reacts to the player swing and direction. When intersect with other objects it destroys them if hit right.

3.4.4 Interfaces: Seen on the screen as first person, controlled by the player.

3.4.5 Data: Unity handles data.

3.5 GameManager

3.5.1 Type: An object

3.5.2 Purpose: manages the points and strikes of the player in the game

3.5.3 Functional Inputs & Outputs: checks if an object hit or missed, then updates points or lives in the game.

3.5.4 Interfaces: Only the game objects interface with the Game Manager, not the player.

3.5.5 Data: GameManager saves the data on memory, during the game, if the player reaches high score it can save the points and name in the database.

4 User Interface Design:

4.1 Main Menu Scene :

4.1.1 Description of the user interface

Here the player can choose if to start new game, to watch Hall Of Fame, set options, read instructions or watch the credits.

4.1.2 Objects and Actions

Object	Action
Start New Game Button	Click
Hall of Fame Button	Click
Options Button	Click
How to Play Button	Click
Credits Button	Click

4.1.3 Screen Image

4.2 Hall Of Fame :

4.2.1 Description of the user interface

Here the player can watch the highscores of all players from all difficulty.

4.2.2 Objects and Actions

Object	Action
Hall of Fame Table	Watch only
Back to Menu Button	Click

4.2.3 Screen Image

4.3 Instructions screen:

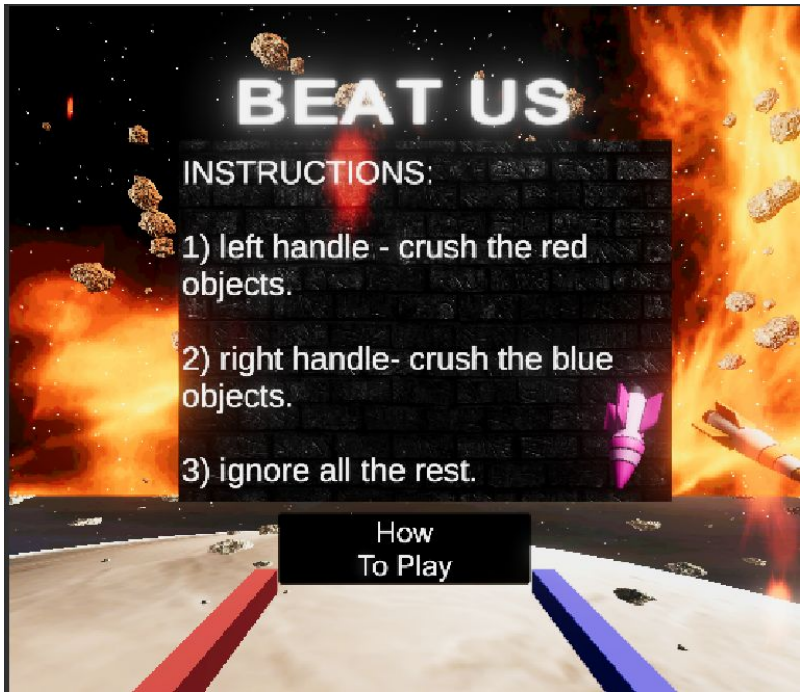
4.3.1 Description of the user interface

Here the player can watch the instructions and learn the rules how to play the game.

4.3.2 Objects and Actions

Object	Action
Instructions text	Watch only
Back to Menu Button	Click

4.3.3 Screen Image



4.4 Options Screen:

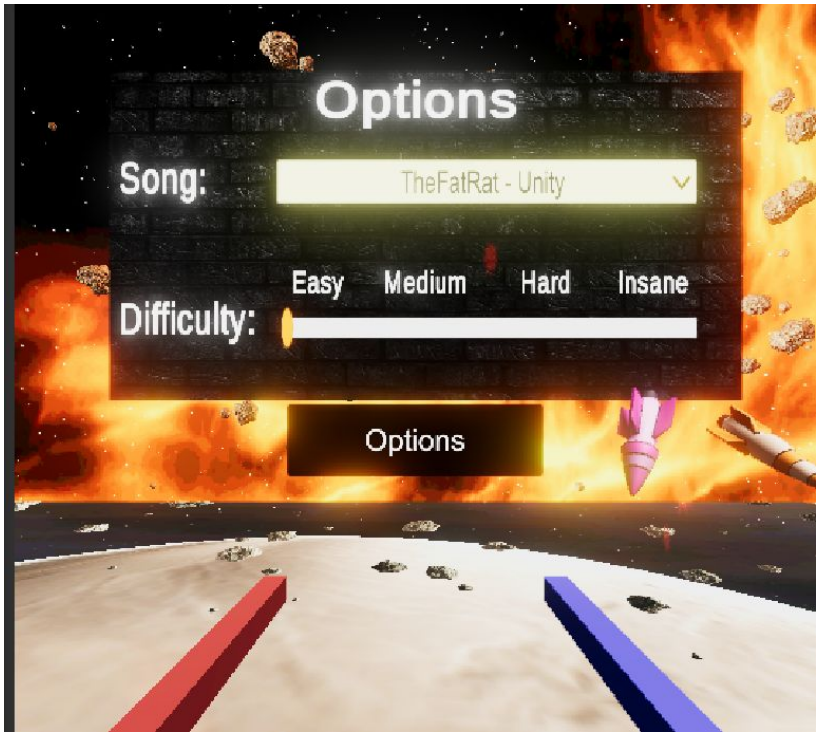
4.4.1 Description of the user interface

Here the player can set the difficulty and to pick a song to play with.

4.4.2 Objects and Actions

Object	Action
Songs Drop Table	Pick one
Difficulty track bar	Pick one
Back to Menu Button	Click

4.4.3 Screen Image



4.5 Game Screen :

4.5.1 Description of the user interface

This is actually the game we developed. Here the player will play the game and his the red/blue cubes with his sabers.

4.5.2 Objects and Actions

Object	Action
Right Bat handle	swing
Left Bat handle	swing
Strikes counter	accumulate
Cubes	thrown at the player
Points counter	accumulate

4.5.3 Screen Image

Software Test Documentation (STD):

1. Introduction:

In this part we will describe the tests that we will performed in the system to ensure compliance with the requirements and objectives we have set. We will set the test and how to do it, as well as the expected result.

2. QA and Sanity Test:

a. Features to be tested

- Test Integration with Oculus headset and handles
- Test responsiveness of the handles with the screen bats.
- Test the objects running towards the player in a changing rhythm.
- Test the that the database is well interacted with the game.

b. Features not to be tested

- Test the game on VR Headset that are not Oculus Rift's brand.
- Test the game with "weak computers" which has the minimum requirements to play the game but not properly.
- Test the Oculus Rift's performances and limitations.

c. Environmental needs

- Oculus Rift - VR headset and handles.
- Unity engine must be installed on the computer with minimum version 2019.1.1.
- PC or laptop that has at least the minimum requirements and system specifications for Oculus Rift S - can be found in the following link: https://support.oculus.com/248749509016567/

d. Main Test Cases

2.4.1 Case – 1:

Options button - pick a song.

2.4.1.1 Purpose:

Check if song that had been selected in the Options will be played in the game.

2.4.1.2 Inputs

2.4.1.2.1 Selected song from list.

2.4.1.3 Expected Outputs & Pass / Fail Criteria:

2.4.1.3.1 The song will be played in the game.

2.4.1.4 Test Procedure:

2.4.1.4.1 The game is starting and Main-Menu shown.

2.4.1.4.2 The player click on the Options button.

2.4.1.4.3 The Options screen opened.

2.4.1.4.4 The player selects a song from the list.

2.4.1.4.5 The player close the Options screen by click the Options button.

2.4.1.4.6 The Main Menu scene will shown.

2.4.1.4.7 The player will select to start a new game.

2.4.1.4.8 The game will start with the selected song playing in background.

2.4.2 Case – 2:

Watch Instructions.

2.4.2.1 Purpose:

Check the Instructions screen.

2.4.2.2 Inputs - קלטים:

2.4.2.2.1 Click on “How To Play” button.

2.4.2.3 Expected Outputs & Pass / Fail Criteria:

2.4.2.3.1 The Instructions screen opened.

2.4.2.4 Test Procedure:

2.4.2.4.1 The game is starting and Main-Menu shown.

2.4.2.4.2 The player click on “How To Play” button.

2.4.2.4.3 The Instructions screen opened.

2.4.2.4.4 The Instructions screen will include the instructions to the game and the player can read it.

2.4.2.4.5 The player clicks the Instruction button.

2.4.2.4.6 The Instructions screen closed and main menu shown.

2.4.3 Case – 3:

Watch Hall Of Fame.

2.4.3.1 Purpose:

Check the Hall Of Fame scene.

2.4.3.2 Inputs - קלטים:

2.4.3.2.1 Click on “Hall Of Fame” button.

2.4.3.3 Expected Outputs & Pass / Fail Criteria:

2.4.3.3.1 The Highscore table will be shown.

2.4.2.4 Test Procedure:

2.4.3.4.1 The game is starting and Main-Menu shown.

2.4.3.4.2 The player click on “Hall Of Fame” button.

2.4.3.4.3 The Hall Of Fame scene opened.

2.4.3.4.4 The player can watch the highscores.

2.4.3.4.5 The High Scores table is sorted by the scores.

2.4.3.4.6 There is 0-10 scores in the table.

2.4.3.4.7 The player clicks the “MENU” button.

2.4.3.4.8 The “Main Menu” scene shown.

2.4.3 Case – 4:

Bats hit target objects

2.4.4.1 Purpose:

Check if the bats hit an object and the game reacts to it.

2.4.4.2 Inputs - קלטים:

2.4.4.2.1 Swinging bats at object (A cube).

2.4.4.3 Expected Outputs & Pass / Fail Criteria:

2.4.4.3.1 The live score will be raise by 100 points.

2.4.2.4 Test Procedure:

2.4.4.4.1 The game is starting and Main-Menu shown.

2.4.4.4.2 The player click on “New Game” button.

2.4.4.4.3 The game will start.

2.4.4.4.4 The system spawn cubes that coming forward to player.

2.4.4.4.5 The player will hit the red cube with his red saber by the arrow.

2.4.4.4.6 The cube will be crushed and disappear.

2.4.4.4.7 The live score will be raise by 100 points.

2.4.5 Case – 5:

Finish game.

2.4.5.1 Purpose:

Check the finish game screen.

2.4.5.2 Inputs - קלטים:

2.4.5.2.1 Finish to play a game.

2.4.5.3 Expected Outputs & Pass / Fail Criteria:

2.4.3.3.1 The Finish screen will show with the following details: score, hits, combos, button of Menu and restart. In case of achieving new highscore: Name field and “Save Highscore” button.

2.4.5.4 Test Procedure:

2.4.5.4.1 The game is starting and Main-Menu shown.

2.4.5.4.2 The player click on “New Game” button.

2.4.5.4.3 The game starting.

2.4.5.4.4 The system spawn cubes that coming forward to player.

2.4.5.4.5 The player plays the game.

2.4.5.4.6 The song was over.

2.4.5.4.7 There is no more cubes in the screen.

2.4.5.4.8 The Live score in the top left screen disappeared.

2.4.5.4.9 The FInish screen shown.

2.4.5.4.10 The FInish screen will show the score,hits and combos of the last game.

2.4.5.4.11 The player clicks the Restart button.

2.4.5.4.12 The game will restart and play the same song.

2.4.6 Case – 6:

Game Over

2.4.6.1 Purpose:

Check the Game Over working well.

2.4.6.2 Inputs - קלטים:

2.4.6.2.1 Play the game without hit the objects.

2.4.6.3 Expected Outputs & Pass / Fail Criteria:

2.4.6.3.1 The Finish screen will show after 5 missed objects with “Game Over” title and will play a sound of “GAME OVER”.

2.4.6.4 Test Procedure:

2.4.6.4.1 The game is starting and Main-Menu shown.

2.4.6.4.2 The player click on “New Game” button.

2.4.6.4.3 The game starting.

2.4.6.4.4 The system spawn cubes that coming forward to player.

2.4.6.4.5 The player ignore the cubes and doesn't do anything.

2.4.6.4.6 The system will add “X” in the top left, under the score when each cube hit the player body.

2.4.6.4.7 The system will update the score top left with 80 points less.

2.4.6.4.8 The Finish screen will show when the 5 “X” will appear.

2.4.6.4.9 The system will stop the spawn of objects and all the objects disappear.
2.4.6.4.11 The system will stop the song.
2.4.6.4.12 The system will play a sound of “Game Over”.
2.4.6.4.13 The Finish title will be “Game Over”.
2.4.6.4.14 The Score will be -400.

2.4.6.4.15 The Combos will be 0.

2.4.6.4.16 The Hits will be 0.

2.4.6.4.17 The player will select the Menu Button.

2.4.6.4.18 The Main Menu scene will be shown.

2.4.7 Case – 7:

Save High Score

2.4.7.1 Purpose:

Check the add score to High Scores.

2.4.7.2 Inputs - קלטים:

2.4.7.2.1 Play the game.

2.4.7.3 Expected Outputs & Pass / Fail Criteria:

2.4.7.3.1 The High Score table will update with the player score.

2.4.7.4 Test Procedure:

2.4.7.4.1 The game is starting and Main-Menu shown.

2.4.7.4.2 The player clicks on “New Game” button.

2.4.7.4.3 The game starting.

2.4.7.4.4 The system spawn cubes that coming forward to player.

2.4.7.4.5 The player plays according to the rules to gain maximum points.

2.4.7.4.6 The system will raise the score according to the player hits.

2.4.7.4.7 The song was over.

2.4.7.4.8 There is no more cubes in the screen.

2.4.7.4.9 The Live score not shown anymore.

2.4.7.4.10 The Finish Screen shown with title: “HIGH SCORE”

2.4.7.4.11 There is fireworks on the screen and player can hear it.

2.4.7.4.12 The Score is positive number.

2.4.7.4.13 The rest of the numeric fields is positive.

2.4.7.4.14 There is “Save High Score” button exist in the screen.

2.4.7.4.15 The player clicks the “Save High Score” button without fill name.

2.4.7.4.16 There is no change in screen and “wrong” sound play.

2.4.7.4.17 The player type his name in the Name field.

2.4.7.4.18 The player clicks the “Save High Score” button.

2.4.7.4.19 Confirmation sound played.

2.4.7.4.20 The “Save High Score” button is disable.

2.4.7.4.21 The player clicks the “Menu” button.

2.4.7.4.22 The “Main Menu” scene shown.

2.4.7.4.23 The player clicks the “Hall Of Fame” button.

2.4.7.4.24 The player clicks the “Save High Score” button.

2.4.7.4.25 The player’s High Score shown once only with his name and score.

2.4.7.4.26 The High Score table is sorted by scores..

3. Functional & Usability Tests

We will form test users group and let them play the game.

We will ask them if they understand and enjoy to play the game.

3.2:

We will check the indices that we set in the sow.

4. Schedule

Test	Start Date	End Date	Duration
Options button - pick a song	7.5.20	7.5.20	1 day
Watch Instructions	7.5.20	7.5.20	1 day
Watch Hall of Fame	7.5.20	7.5.20	1 day
Bats hit target objects	7.5.20	10.5.20	3 days
Check the finish game screen	8.5.20	10.5.20	2 days
Check the Game Over working	10.5.20	10.5.20	1 day
Save High Score	11.5.20	12.5.20	2 days
Usability Testing	12.5.20	15.5.20	4 days