Technical University of Denmark

02515 Health Care Technology

Workout Memory Game

Authors:

Andreas Hallberg Kjeldsen
s092638@student.dtu.dk

Morten Chabert Eskesen
s133304@student.dtu.dk

December 3, 2014

Abstract

Overweight and obesity is an increasing problem in Denmark. Being physically active can prevent obesity and in turn make a person more healthy. We have created a game that requires the player to be active, by performing a sequence of exercises. The exercises must be performed in the right order and within a certain timeframe. The game has been build using Unity 3D and the Microsoft Kinect Sensor. The Kinect is used to track the player and to confirm the exercises are performed satisfyingly. We have tested our game on two group of kids in the age between 12 and 15. The tests showed an average increase in heart rate of 52.9%.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Formalities

This report is a result of a project done in the course *Health Care Technology* taken at DTU. The project has been a collaboration between us both, Andreas and Morten, where we each have been equally responsible for all parts of the project. Andreas' mother works at an elementary school and by virtue of that we were able to test the game on kids from elementary school.

1.2 Objectives

The objective and product of the course Health Care Technology is to create a health-promoting interactive game by using Unity and Kinect technology. The game should solve a health care problem in Denmark. The course puts a lot of focus on health care challenges in today's Denmark and how the challenges can be solved by modern technology. In the future we will rely more and more on modern technology for health care problem as one of the problems in health care is the increasing need for treatment of the aging population. Another problem is the number of obese people in Denmark which has increased considerably in the last decade [1].
This project aims to aid the solving of the increasing obesity problem in Denmark by creating a game that will increase the physical activity of people of age 13 and up, such that obesity can be prevented and obese people can become more healthy.

1.3 Overweight & obesity

Overweight and obesity are defined as abnormal or excessive fat accumulation that presents a risk to health [2]. An international measure known as Body Mass Index, henceforth *BMI*,

is commonly used to classify people as underweight, overweight and obese. BMI values are age-independent and the same for both sexes [3]. BMI is calculated by the following:

$$\text{BMI} = \frac{\text{Weight (kg)}}{\text{Height}^2\text{(m)}}$$

When calculating this you can use it to see if you are classified as underweight, normal, overweight or obese by the BMI scale. The BMI classifications and their corresponding values are defined as follows [3]:

Table 1.1: BMI classification

Classification	**BMI**
Underweight	Below 18.5
Normal range	18.5 - 24.99
Overweight	Over 25
Obese	Over 30

Overweight and obesity are major risk factors for a number of chronic diseases. These diseases include diabetes, cardiovascular diseases and cancer. In Denmark the total expenses linked with obesity and the treatment thereof on hospital was calculated as being 1.1 billion kroner in 2004 [4].

1.4 Memory

There exists various ways to improving ones memory. One of the techniques is to exercise regularly. Many studies have shown that the parts of the brain that control memory and thinking have greater volume in people who exercise versus people who do not exercise [11]. A study from University of California, Los Angeles showed that doing brain training games can significantly improve memory and language skills - especially in older people. The brain training programs can also delay dementia in later life [10]. Although the study used many different brain training games with different purposes, like reasoning, short- and long-term memory etc., we have decided to combine a brain training game with exercise. This has resulted in a game that combines a memory game with exercise.

Chapter 2

Game details

2.1 Game description

The Workout Memory Game, as the name suggests, is a game where both memory and physical exercises are key components. The game consists of rounds, where in each round you have to perform certain exercises. The exercises are Squat and Jump. Round 1 consists of 1 exercise, Round 2 consists of 2 exercises - the previous one and a new exercise chosen at random and the game continues like this until game over.

2.2 Requirements specification

The focus of the game is to get people to exercise, hence it's vital to incorporate a requirement of movement. Though it might seem redundant, it is also important to make requirements on the visualization of the game, the difficulty of the game and the usability of the game.

2.2.1 Game setup

The game should be easy to setup. Hence there should be no additional software installations required to get the program working. There might be a small setup required for the actual room the game is to be played in. As this is a game that requires physical activity, the player should be able to move around, jump up and don and squat without hitting any obstacles. To enhance the gaming experience, the lights in the room should be turned on.

2.2.2 Game start

The player must be able to start the game just by interacting with the Kinect. When a game is finished, the player should be able to restart the game without having to first close down the game and then run it again.

2.2.3 Player recognition

The game must be able to recognize players[1]. The game must support one player, though two player mode would be nice to have. The game should support players of various height, width and depth, thus the game should be able to calibrate when a new player is recognized. The game should be able to detect if a player is missing and pause the game accordingly, though this should only be in effect between rounds to prevent cheating.

2.2.4 Game progression

While playing the game, the user should be able to determine if they're progressing or not. The game should consist of rounds, which the player can use to determine if they're progressing or not. The player should also be rewarded when progressing in the game. The reward should be points which in turn should increase the players score.

2.2.5 Exercises

The exercises the player has to perform should be picked at random. Picking the exercises at random forces the player to be paying attention and to exercise their memory. Each round a new exercise must be added to the sequence of exercises to perform. The order of the exercises must not be changed within a single game. The exercises are shown sequentially. The position on the screen where the exercise will be shown should be chosen at random the first time, in the following rounds the exercise should be shown at the same position.

2.2.6 Gesture recognition

The basis for the game, is that the player should perform a specific set of exercises. The game must be able to recognize these exercises. The game should also be able to recognize exercises that are being performed in a consecutive manner. The recognition should be flexible, in such a way that false positives for gestures are very rare and preferably not happening at all.

2.2.7 Timing

Each exercise should be performed within a set amount of time. When an exercise is performed correctly, the timing should be reset and allow the same amount of time for the next exercise. The amount of time should allow for the player to remember which exercise to perform and for the player to actually perform it. The amount of time should not allow for the player to write down the exercises, look up which to perform or become too relaxed.

[1] To recognize a player, means identify that a player is in front of the Kinect. It does not mean identifying a specific person.

2.2.8 Scoring

When the player is rewarded points for performing the right exercise, the amount of points being rewarded along with the exercise the user performed, should be shown on the screen. Showing the rewarded points and the exercise is a way for the game to provide feedback for the player. The feedback informs the player that the exercise was recognized and that they should progress onto the next exercise. The feedback should be shown on the screen where the exercise perform was shown. The amount of points gained should depend on how far the player has progressed in the game and how quickly they perform the exercise.

2.2.9 Game over

The game should end if an exercise is not performed within the allowed time or if the wrong exercise is performed. The player must be informed that the game is over. A summary of the game should be displayed to the player.

2.2.10 Summary

Below is a summary of the requirements specification modeled as a table. Each requirement states two scenarios, the *need to have* and the *nice to have*. As a minimum the game should support the need to haves, but we aim for the game to support most, if not all, of the nice to haves.

Table 2.1: Requirement specification, describing the minimum requirements (Need) and the desired requirements (Nice).

Requirement	Need	Nice
Room setup	A well lit room with no objects within the viewport of the Kinect	A room with some lights turned on, doesn't matter if there are objects within the viewport of the Kinect
Game start	The player must able to start the game by interacting with the Kinect	The player must be able to both start and restart the game by interacting with the Kinect
Amount of players	One player is supported	Two players are supported
Player recognition	The game should be able to recognize the player, calibration poses might be required	The game should be able to recognize the player, calibration is done automatically

Continued on next page

Table 2.1 – continued from previous page

Requirement	Need	Nice
Player missing	The game should be able to detect if a player is missing and pause the game accordingly	The game should be able to detect if a player is missing and pause the game accordingly, but only between rounds, to prevent cheating
Game progression	The game consists of rounds which the player can progress though	The game consists of rounds which the player can progress through, additionally points are also rewarded
Exercises	New exercise each round while keeping the order of previous exercises, the exercises are all shown in the middle of the screen	New exercise each round while keeping the order of previous exercises, the exercises are shown at random positions
Gesture recognition	The gestures are recognized, false positives are rare	The gestures are recognized, false positives are very rare if not happening at all
Timing	Timing of exercise, player has plenty of time between exercises	Timing of exercises, player has little time between exercises and must remain active and observant.
Scoring	Points are earned when performing the right exercise	Points are earned when performing the right exercise, amount of points are based on game progression
Feedback	The player gets visual feedback when an exercise has been performed	The player gets visual feedback when an exercise has been performed while also being notified about the amount of points earned
Game over	The game can end when the player fails to perform the right exercise within the allowed time	The game can end when the player fails to perform the right exercise within the allowed time, the obtained score is also shown on the screen

2.3 Technology

The game has been developed using Unity 3D and the Microsoft Kinect Sensor technology.

2.3.1 Unity 3D

Unity 3D, henceforth Unity, is a Game Engine that can be used for developing games in both 2D and 3D. Unity includes a physics engine. In the Unity Asset Store, an enormous amount of community created assets are available, some have to be bought, others are available for free. The assets range from complete games to single assets with very specific functionality. Using the assets from the Asset Store can speed up the development process and also makes it possible to create good looking and functional games without skills in both design and coding.

2.3.2 Microsft Kinect Sensor

The Microsoft Kinect Sensor technology is capable of tracking human movements, by tracking a specific set of joints of the humanoid skeleton, see Figure 2.1. Specific limbs can also be tracked by using two or more tracked joints and calculating the movement of the limbs based on the joints.

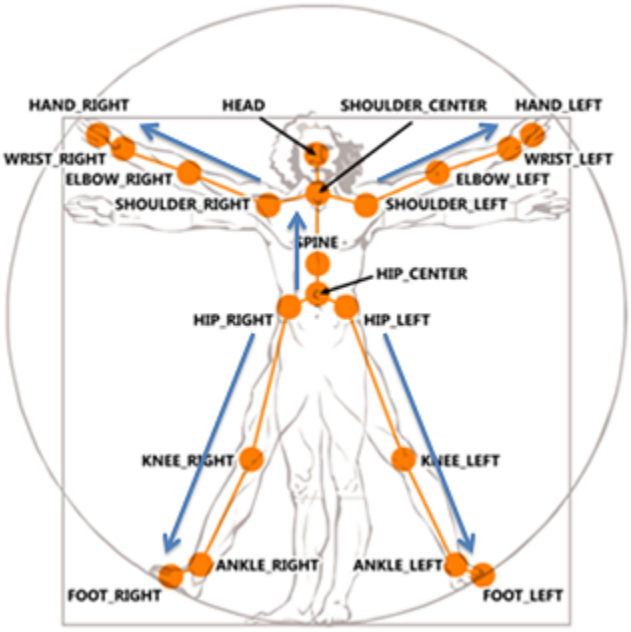

Figure 2.1: The joints trackable by the Microsoft Kinect Sensor technology [13].

Chapter 3

Implementation

The game has been implemented using Unity3D Free v4.5.4 as the game engine, the Visual Studio IDE for coding in C# and the Microsoft Kinect Sensor v1 for player tracking. The game has been made in an incremental manner, using some of the principals of Scrum. Git and Github has been used for source control, feature discussion and issue tracking.

3.1 Scene

The scene was kept simple. There's a skybox with a solid color used as the background for the game. GUIText objects are then used to display various texts. The game is to be kept simple and not overly fancy, therefore the texts are using a font with the look and feel of good old 8bit games. The font is readable close by and when it's scaled up. The font is called *Press Start 2P* and is Open Font licensed (OFL) [5].

3.2 Game management

A set of classes were made to manage the game, these include a *Brain* script, *GameEvent-Manager* script and *GUIManager* script. These scripts make up the primary game functionality. The Brain script is in charge of starting a new game, creating new rounds, adding moves to perform while playing, keeping score and ending the game. The GUIManager is in charge of showing/hiding and updating the GUITexts used within the game. The GameEventManager is used for delegating messages about the state of the game (start game, end game, new round).

3.3 Keyboard input

The game can be played using the keyboard. This was implemented for the sake of testing the game management functionality without having to interact with the Kinect. Pressing

the up arrow key indicates a Jump, pressing the down arrow key indicates a Squat and pressing the space bar indicates a new game is to be started.

3.4 Kinect integration

The Kinect had to be hooked up to the game. At first the KUInterface code that was handed out in the course was used to interact with the Kinect. It worked, but was missing some functionality that was required, hence another wrapper for interacting with the Kinect was found on the Unity Asset Store [6]. The other wrapper included a KinectWrapper class that is in charge of communicating with the Kinect using PInvoke calls and a KinectManager class to be used with Unity. The KinectManager included a large amount of code for gesture recognition that was overly complicated, hence the code has been removed together with several options of the KinectManager that was of no use.

3.5 Joint tracking

The KinectManager provides functionality for obtaining the current positions of the joints in *Viewport Space*. To verify that the joints are tracked, GameObjects were setup to follow the joints. GameObjects use *World Space*, hence the joint position had to be changed into World Space. The calculations required for transforming the positions were included with the KinectManager [6]. A class for tracking all the joints and overlaying them with GameObjects were made, further a helper class for obtaining, getting and setting the joint positions were made, this also allowed for logging of the joint positions to a log file.

3.6 Gesture recognition

The joints are tracked in Viewport Space, which is a 2 dimensional space and goes from -1.0 to 1.0 in both the x and the y direction. There are no certainties that the joints are always at the same height when the player is idle, i.e. the *HipCenter* joint is not always located at $y = 0.0$. The joint position logging was then used to find a base interval in which it could be assumed that the HipCenter joint was located when the player is idle. Same procedure for the other joints used in the jump gesture (KneeLeft, KneeRight), the squat gesture (KneeLeft, KneeRight, HipLeft, HipRight) and the clap gesture (HandLeft, HandRight). To actually be a gesture, the player had to perform a specific action. When jumping, the player have to raise their knees a certain amount, when squatting the player has to lower their knees and hips a certain amount and when clapping the player has to move their hands together. The amount the joints had to move was also found using the joint positions log. By performing the exercises while logging the joint positions, the goal intervals in which it could be assumed the player would have reached the required pose was found. Using the base intervals and the goal intervals, the amount required to

move the joints were found. Detecting a gesture was then just the matter of tracking the joints required in the gesture over time, noting if the joints moved by the required amount. Restrictions are in place, stating that the player has to be within the base intervals for the joints before a gesture is tracked. This is e.g. to prevent two simultaneous squats to be perceived as a jump.

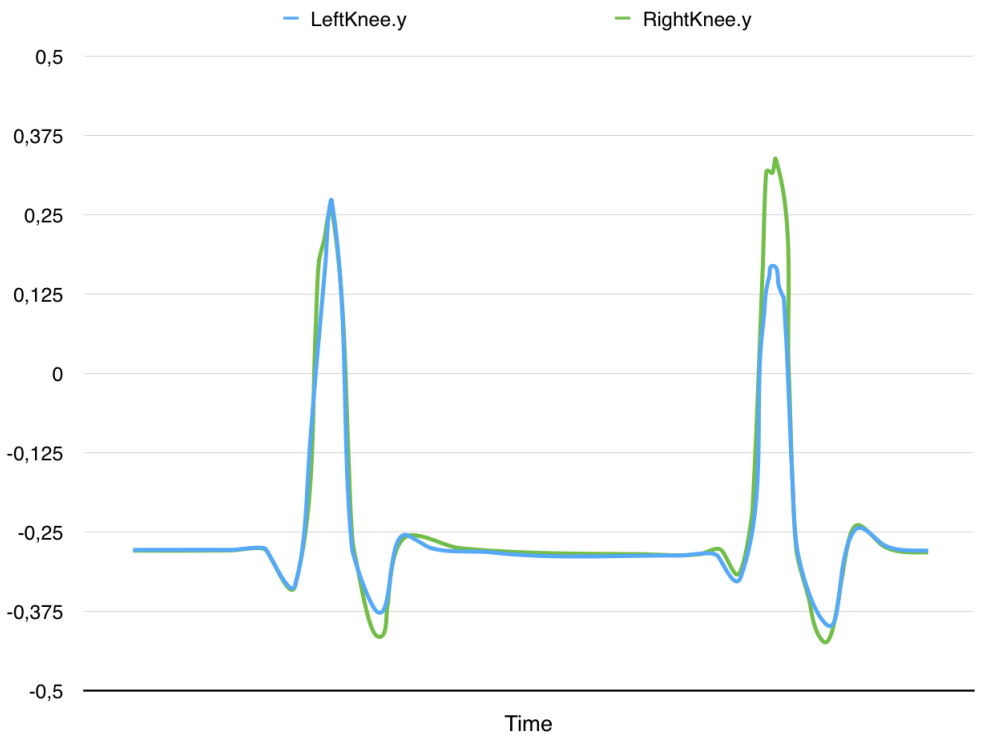

Figure 3.1: Graph showing the positions of the joints KneeLeft and KneeRight during a session where the player jumped.

3.7 Visual improvements

The primary visual used in the game is the feed from the RGB camera of the Kinect. The texts used was clear and readable when just testing the Game Management, as the background was a solid color and the texts was another solid color with great contrast. When using the feed from the RGB camera as the background of the game, there are no limits to the colors, shapes and lightning. Therefore a border was added to the text. In

Unity 3D Free v4.5.4 the GUIText component does not have a border feature. To fix this, a script was made mimics a border around the text, by drawing the text multiple times behind the actual GUIText. While developing the game, a preview screen is visible that shows what the game will look like. Everything seems fine in the preview, but when the game is launched and played in full screen mode, the texts weren't scaling, hence they were quite hard to read from a distance. Another script was created that scales the texts. A third script was created, which makes the texts fade out after a certain amount of time.

Chapter 4

Test

This chapter will outline our test specification and the test results from testing on children from an elementary school. Lastly it will discuss the test results in regards to the test specification.

4.1 Test specification

In this section we have modeled our test specification as a table. Each test states two scenarios, the *need* to have and the *nice* to have. The needs to have describe the minimum acceptance criteria and the nice to have describe the desired acceptance criteria.

Table 4.1: Test specifcation, describing the minimum acceptance criteria (Need) and the desired acceptance criteria (Nice).

ID	Test	Need	Nice
T1	Start & setup time	1 min	30 seconds
T2	User interface	User is instructed and can play	User can play without instructions
T3	Game progression	Game progression is reported	Difficulty of game is adapted to how far the player has progressed in the game
T4	Jump recognition	A jump is recognized	Height of a jump is recognized
T5	Squat recognition	A squat is recognized	A deep squat is recognized

Continued on next page

Table 4.1 – continued from previous page

ID	Test	Need	Nice
T6	Clap recognition	A single clap is recognized	A double clap is recognized
T7	Calibration	User has to stand in specific pose to calibrate	Calibration is done automatically
T8	Player's heart rate	20% increase in heart rate of player on average	40% increase in heart rate of player on average
T9	Player's memory	The player at least stays at the same round	The player progresses at least 1 round, as long as their physique allows it
T10	Missing player	If no player is detected, the game should pause	If no player is detected, the game should pause and notify about the player missing

4.2 Test results

Besides testing on ourselves, our friends and family, we also wanted to take the game out in the world and apply it in an environment where it could possibly have a positive effect. Our game was tested on kids in age range 12-15 at an elementary school in Helsingør. The game was tested in a classroom, hence we could test if furniture in the background would interfere. We found that the player had to stand at least 12 meters away and be right beside some furniture, before it would have an effect on the Kinect registering the joints properly. One of the measurable features of a training game is the heart rate of the player. When testing we noted the heart rate of the player before and after playing the game. What follows is a table of the test results obtaining from the elementary school.

Players occurring with the same name are not different players but the same player's second try. The first number in the heart rate column is the heart rate before and the second number is the heart rate after - i.e. 76 $\rightarrow$ 114 means a heart rate of 76 before and 114 after playing. The percentage describes how much the heart rate increased in percentage. FR is Failure Reason where the following codes apply:

1 Wrong exercise done by the player

2 Time ran out, i.e. no exercise was done in time

3 Registration error

4 Computer crashed

Table 4.2: Results after testing 14 kids with an average age of 13.9. The results show an average increase in heart rate of 52.9 %.

Name	**Age**	**Gender**	**Heart rate**	**Diff.**	**%**	**Round**	**Score**	**FR**
Amine	15	F	76 → 114	38	50.0	5	2264	1
Anders	15	M	64 → 93	29	45.3	3	253	1
Anders	15	M	67 → 102	35	52.2	4	1057	1
Ditte	14	F	69 → 117	48	70.0	9	10045	4
Ditte	14	F	74 → 141	67	90.5	12	27497	1
Frederik	15	M	73 → 94	21	28.8	3	342	2
Freja	13	F	72 → 80	8	11.1	2	58	1
Freja	13	F	77 → 96	19	24.7	3	143	3
Imon	15	M	78 → 100	22	28.2	5	1971	1
Isabella	15	F	73 → 120	47	64.4	7	6132	1
Jasper	15	M	71 → 87	16	22.5	3	652	1
Jasper	15	M	73 → 123	50	68.5	7	6129	1
Lærke	12	F	79 → 106	27	34.2	4	1077	3
Lærke	12	F	73 → 121	48	65.8	11	18528	1
Marcus	12	M	76 → 114	41	54.0	4	3308	1
Marcus	12	M	73 → 132	59	80.8	13	36939	1
Nicklas	15	M	64 → 104	40	62.5	5	1983	1
Nicklas	15	M	73 → 114	41	56.2	8	6931	1
Philip	13	M	73 → 127	54	74.0	9	10116	1
Philip	13	M	78 → 141	63	80.8	12	22991	1
Siw	15	F	74 → 123	49	66.2	5	1411	2
Siw	15	F	72 → 129	57	79.2	12	19345	1
Zenah	13	F	74 → 91	17	23.0	4	1304	1
Zenah	13	F	83 → 112	29	34.9	8	9775	1
Average	13.9				52.9	6.9	7927.1	

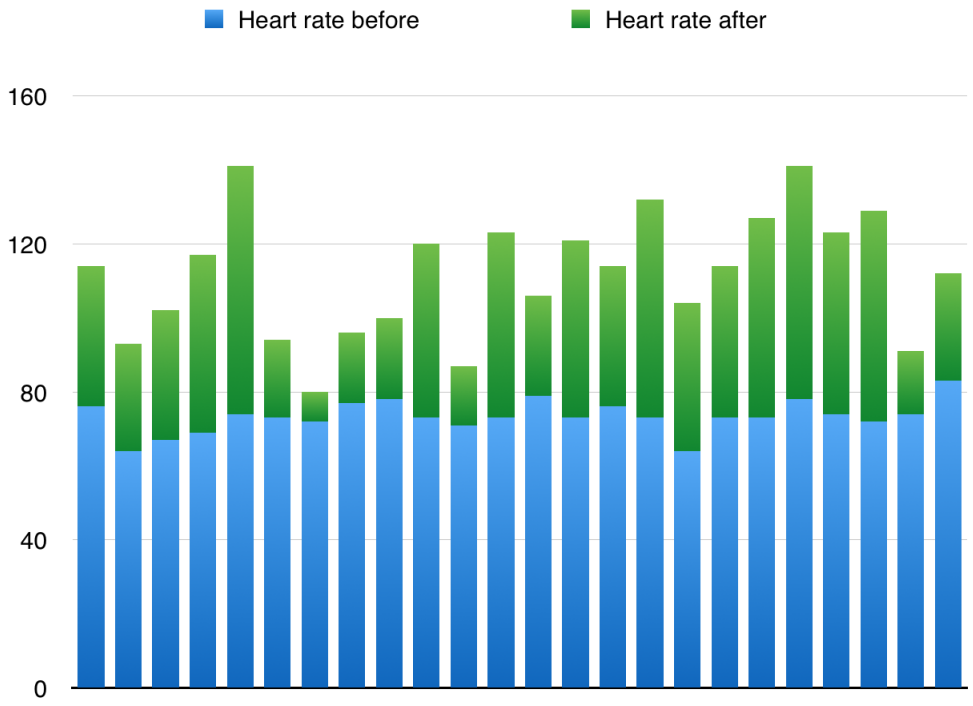

Figure 4.1: Graphical overview of the test results

4.2.1 Feedback

The children playing our game gave a lot of feedback to the game. As a spectator you could very clearly see that they were trying hard to remember the exercises especially in the higher rounds. What follows is a list of feedback given by the children.

- It is physically exhausting
- You could clearly see when an exercise was done correctly by virtue of the green text on the screen with the points gained
- It is hard to remember the order of exercises, especially when the exercises alternate very often
- It is like using a Jump Rope (said by a girl who had many jumps)
- At first it was difficult to tell how high to jump and how low to squat in order for the game to register the exercise properly

Figure 4.2: Pictures from the test session

4.3 Discussion

Based on the test results, the following discussion will analyze to which extend the tests in the test specification have been passed.

T1: Start & setup time

The test session showed that the game was never more than 30 seconds to startup and be ready for playing. This means that we have achieved the **nice** requirement of this test case. However the only reason the startup took some time was because of the Kinect having to be started and therefore we would have liked a more smooth transition such as turning on a PlayStation so the screen is not all black when starting up.

T2: User interface

During the test session the users were given a short introduction, explaining they had to jump if the game said jump and squat if the game said squat. Even without further

introduction, it was clear that the users quickly understood how the gesture recognition feedback worked, how high to jump and how deep to squat. The game states the commands to perform but may lack further explanation of when and how to jump and squat. The test case only satisfies the **need** requirement.

T3: Game progression

The requirements for the gestures are not becoming more strict the further the player has progressed. But the amount of exercises to remember is increased, so in a sense, the difficulty of the game is adapting to the performance of the player. Further it could be introduced that the previous exercises are either not shown again or the time they are shown is decreased. The test case only satisfies the **need** requirement.

T4: Jump recognition

If the height of a jump had been recognized we could have logged the heights of all the jumps made by players. We could have used this to make an analysis of how high each player jumps and who had the record among the test subjects. As mentioned the requirements for jumping are not becoming more strict as the game is progressing. We would have liked to have done this in order to make the game more difficult as the game progressed such that the player had to jump higher and higher. The test case satisfies the **need** requirement.

T5: Squat recognition

Requiring that the player was to do a deep squat was definitely possible code-wise, but during our test session, it showed that not all people are flexible enough to actually do a deep squat, thus the requirement for the squat was slacked a bit. With the requirement slacked, the player can still perform a deep squat and it will still be recognized. Encouragement towards performing the exercises to their full extent should be a part of the introduction to the game. The test case satisfies the **nice** requirement.

T6: Clap recognition

The test showed that a single clap to start the game could be confusing. It is not uncommon for people to put their hands together, just as if they would clap. Requiring a double clap proved to be more appropriate, as it had to be a conscious action performed by the player. The test case satisfies the **nice** requirement.

T7: Calibration

The test session showed that we did not have to restart the game between each player or position each player in a specific pose in order to calibrate. The calibration was done automatically. The test case satisfies the **nice** requirement.

T8: Player's heart rate

The test session yielded a result that showed an average of 52.9% increase in heart rate. This is a quite high increase in heart rate for a game. It has to be taken into account that the test subjects were quite excitable and very competitive with their friends and laughing while playing. Therefore we can not point out the game as the single factor for the increase in heart rate. However the test case satisfies the **nice** requirement which was a 40% increase in heart rate.

T9: Player's memory

The test session showed that each player who tried the game more than once progressed at least 1 more round the second time than their first time. It means that the players are getting better and better at the game and thereby doing more exercises and increasing their heart rate and their ability to remember the exercises to perform. The test case satisfies the **nice** requirement.

T10: Missing player

The test session showed that each time a player was not able to be registered then the game notified that a missing player was missing and paused the game. This was however only done between rounds, i.e. before exercises for the round is being shown on the screen in order to prevent players from cheating by writing down the order of exercises. The test case satisfies the **nice** requirement.

Chapter 5

Conclusion & Discussion

This chapter will conclude and discuss the work done in this project. It will also discuss the possibilities for further development of the game.

5.1 Conclusion

In this project we have used Unity and Kinect technology to create a game that challenges both the memory and the fitness of the player. The game combines the two physical exercises Jump and Squat to create a Workout Memory Game. The game consists of rounds. Round 1 consists of 1 exercise, round 2 consists of the previous exercise and a new exercise and so on until the game is over. The exercises are chosen at random. The player has 5 seconds to perform each exercise and the game ends when a player runs out of time to do an exercise or doing the wrong exercise. The game has fulfilled all the desired requirements from the requirement specification excluding the desired requirement about two player mode. The test results showed an average of 52.9% increase in heart rate, which is more than our initial desired acceptance criteria. The test results also showed that the players also progressed further in the game for each time they played the game.

5.2 Further development

The vision for the game is to extend it into a suite of small games to help the player exercise. The Workout Memory Game should just be one of the small games included in the suite. The other small games could then focus on something else like the arms, suppleness or reaction time. Introducing two player mode would also be a possibility, this way the player not only competes with themselves but also a friend (or foe). As a suite, the players would have a profile where they stats would be logged. The player should then be able to track their progress in the various small games. Based on the player progression, the games should become either harder or easier to allow for further improvement. Ideally the suite could

be linked with various apps like Endomondo and gadgets like the Nike FuelBand to collect even more data, that would be used for player statistics and progression overviews. Visually the suite would also need an overhaul and have a proper interface for navigating the system and exercises. As the game is implemented now each exercise to perform in every round is shown for 2 seconds. In order to introduce more difficulty this could be changed such that only the exercise introduced in the current round was shown for 2 seconds and the others for 1 second. Other possibilities could be that the time exercises are shown decreases as the game is processing.

Bibliography

[1] Sundhedsstyrelsen, *Overvægt* http://sundhedsstyrelsen.dk/da/sundhed/overvaegt

[2] World Health Organization, *Obesity* http://www.who.int/topics/obesity/en/

[3] World Health Organization, *BMI classification* http://apps.who.int/bmi/index.jsp?introPage=intro_3.html

[4] Ministeriet for Sundhed og Forebyggelse, *Samfundsøkonomiske konsekvenser af svær overvægt* http://www.sum.dk/Aktuelt/Nyheder/Forebyggelse/2007/Maj/Samfundsoekonomiske_konsekvenser.aspx

[5] Google Fonts, *Press Start 2P* https://www.google.com/fonts/specimen/Press+Start+2P

[6] RF Solutions, *Kinect with MS-SDK*, v1.11, October 15, 2014 https://www.assetstore.unity3d.com/en/#!/content/7747

[7] P. Murali Doraiswamy and Marc E. Agronin, *Brain Games: Do They Really Work?* http://www.scientificamerican.com/article/brain-games-do-they-really/

[8] Wikipedia, *Memory Improvement* http://en.wikipedia.org/wiki/Memory_improvement

[9] Natural Health, *Do "brain-boosting" computer games improve your memory?* http://www.naturalhealthmag.com/expert-advice/do-brain-boosting-computer-games-improve-your-memory

[10] Victoria Woollaston, *Proof that brain training games DO boost memory and language skills in older people* http://dailym.ai/1yGGhag

[11] Heidi Goodman, *Regular exercise changes the brain to improve memory, thinking skills* http://bit.ly/1ewiOfg

[12] Microsoft Kinect SDK, *JointType Enumeration*, viewed October 2014. `http://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx`

[13] Microsoft Kinect SDK, *Tracking Users with Kinect Skeletal Tracking*, viewed October 2014. `http://msdn.microsoft.com/en-us/library/jj131025.aspx`

Appendix A

This is the first appendix

I think we should avoid putting in all our code...