ACTIVE GAMING USING MICROSOFT KINECT IN SOLVING OBESITY PROBLEM

VINCENT TAN SIN CHIA

UNIVERSITI MALAYSIA PAHANG

ACTIVE GAMING USING MICROSOFT KINECT IN SOLVING OBESITY PROBLEM

VINCENT TAN SIN CHIA

A thesis submitted partial fulfillment of requirements for award of the degree of Bachelor of Computer Science (Computer Graphics And Multimedia Technology) Hons

Faculty of Computer Science and Software Engineering
UNIVERSITI MALAYSIA PAHANG

2014

UNIVERSITI MALAYSIA PAHANG

BORANG PENGESAHAN STATUS TESIS

JUDUL: ACTIVE GAMING USING MICROSOFT KINECT IN SOLVING OBESITY PROBLEM

SESI PENGAJIAN: 2014/2015/1

SAYA VINCENT TAN QIN CHIA (HURUF BESAR)

Mengaku membenarkan tesis/laporan PSM ini disimpan di Perpustakaan Universiti Malaysia Pahang dengan syarat-syarat kegunaan seperti berikut:

1. Tesis/Laporan adalah hakmilik Universiti Malaysia Pahang.
2. Perpustakaan Universiti Malaysia Pahang dibenarkan membuat salinan untuk tujuan pengajian sahaja.
3. Perpustakaan dibenarkan membuat salinan tesis ini sebagai bahan pertukaran antara institut pengajian tinggi.
4. **Sila tandakan (√)

☐	SULIT	(Mengandungi maklumat yang berdarjah keselamatan atau kepentingan Malaysia seperti yang termaktub di dalam AKTA RAHSIA RASMI 1972) *
☐	TERHAD	(Mengandungi maklumat TERHAD yang telah ditentukan oleh organisasi/badan di mana penyelidikan dijalankan) *
✓	TIDAK TERHAD	

[illegible]

Disahkan Oleh

[illegible]

Alamat tetap: 9, PSRN SEKSYEN 6/2W, BDR PUTRA BERTAM 13200 KEPALA BATAS P.P.W.

Tarikh: 7/1/15

Penyelia: ABBAS SALIIMI BIN LOKMAN
Lecturer
Faculty of Computer Systems & Software Engineering
Universiti Malaysia Pahang
Lebuhraya Tun Razak, 26300 Gambang, Kuantan, Pahang.
Tel: 09-549 2423 Fax: 09-549 2144

Tarikh: 7/1/2015

*Sila lampirkan surat daripada pihak berkuasa/organisasi berkenaan dengan menyatakan sekali sebab dan tempoh tesis/laporan ini perlu dikelaskan sebagai SULIT atau TERHAD.

DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged.

15 DECEMBER 2014
CD11080

Vincent Tan Sin Chia

SUPERVISOR'S DECLARATION

"I hereby declare that I have read this report and my opinion this report is sufficient in terms of scope and quality for the submission of PSM 2, Degree in Computer Science (Networking)."

Signature	: [illegible]
Supervisor	:**Abbas Saliimi Lokman**
Date	: 26/12/2014

ACKNOWLEDGEMENT

I would like to express the deepest appreciation to my Final Year Project Supervisor, Mr Abbas Saliimi Lokman, who has shown tremendous attitude and dedication as well as enthusiasm and not to mention his immense knowledge. His valuable guidance and opinion helped me in all the time of research as well as writing of this thesis. Without his supervision and constant help, this thesis would not have been possible.

Besides, I would like to thank my friends, Tan Yi Wen, Wong Di Ing, Tan Wei Ren and Lee Shin Yee for giving me utmost support and care to overcome setbacks and stay focused on the journey to ultimately finish this thesis writing. I greatly value their friendship and I deeply appreciate their belief in me.

Last but not least, I would like to reserve my deepest appreciation to my parents as they are the ones who make this thesis writing possible.

Thank You.

ABSTRACT

Obesity has seen an alarming rate of growth with its widespread influences at both local and global scale trigger a global urgency in solving this problem. In South East Asia, our beloved country Malaysia tops the list for the wrong reason as the fattest country in ASEAN with two of out of five adults are obese. The fattest country in the world is represented by Kuwait with a staggering number of 42.8% of total population are obese. [1] This widespread epidemic is soaring among children and teens. As a result, chronic ailments such as diabetes is becoming an unusual norm in today society. [2] In this research, electronic means are explored in order to provide a feasible solution to counter the rising problem of obesity. The study will cover the utilization of Microsoft Kinect as the viable solution in promoting active gaming among the obesity patients with the assistance of engaging and interactive gameplay that is designed to cater the physical ability of obesity patients. In addition, in this study, Microsoft Kinect would be the primary input that receive gestures performed by user while recognition of gestures will be done in Unity3D game engine as to translate the detected gestures into motion or functionalities inside the designated game. Furthermore, the core gesture advocated in this active game is none other than the running gesture which represents a good form of cardiac workout which is one of the main solution towards obesity other than proper dieting. Besides, in this game, obesity patient or user will get to know the approximate value of calories burned. Lastly, this research is intended to propose a solution in form of an active game that is designed to promote active lifestyle among obesity patients.

ABSTRAK

Obesiti merupakan suatu fenomena negatif yang mengancam kesihatan sejagat dengan rekod penyakit meliputi seluruh dunia. Fenomena ini telah mencetuskan perhatian sejagat untuk menyelesaikan masalah yang semakin memudaratkan kesihatan sedunia. Selain itu, tanah air tercinta Malaysia telah ditempatkan di petak utama antara negara yang paling gemuk di ASEAN dengan statistik menunjukkan bahawa dua daripada lima dewasa di Malaysia dikategorikan sebagai obesiti. Untuk pengetahuan anda, negara yang paling gemuk di dunia ialah Kuwait yang telah mencatat suatu rekod yang mengejutkan dengan sebanyak 42.8% daripada populasi negara tersebut dikategorikan sebagai obesiti. Tambahan pula, obesiti bukan sahaja suatu fenomena yang biasa berlaku kepada dewasa malah penyakit ini telah pun meliputi golongan kanak-kanak. Oleh itu, kejadian bahawa penyakit kencing manis melanda kanak- kanak merupakan suatu perkara ataupun berita yang teramat biasa. Dalam penyelidikan ini, penyelesaian masalah obesiti dengan memanfaatkan alatan elektronik bersama dengan permainan video akan diperkenalkan. Di samping itu, penyelidikan ini akan menumpukan perhatiannya terhadap alatan electronik ataupun sensor keluaran Microsoft yang dikenali sebagai Microsoft Kinect. Penyelidikan tersebut akan memperkenalkan keupayaan Microsoft Kinect dalam menjadi input utama untuk menjayakan penciptaan permainan video yang disesuaikan untuk pesakit obesiti berasaskan keupayaan fizikal pesakit. Tambahan pula, input utama yang digalakkan dalam permainan video ini adalah berasakan penyelesaian utama obesiti yang bertumpu kepada senaman kardio. Contohnya, input utama yang berasaskan senaman kardio merupakan senaman yang melibatkan berlari, berjoging, ataupun berjalan pantas. Akhirnya, dengan menjalani permainan video ini, pesakit obesiti akan dapat mengetahui anggaran kalori yang telah dibakar. Kesimpulannya, penyelidikan ini bertujuan untuk mencadangkan suatu penyelesaian kepada masalah obesiti melalui permainan video untuk mengalakkan gaya hidup yang sihat dan aktif terutamanya dalam kalangan pesakit obesiti.

TABLE OF CONTENTS

LIST OF TABLE

LIST OF FIGURE

LIST OF APPENDICES

CHAPTER1: INTRODUCTION

1.1 INTRODUCTION

Obesity or overweight could be defined as a condition in which excessive accumulation of body fat to an extent that it may result in negative impact on health, leading to reduced life expectancy. People are considered obese when their body mass index (BMI), a formula or measurement obtained by dividing a person's weight by the square of person's height, is greater than $30 kg/m^2$.[3]

The main reason behind obesity is the result of surplus energy stored as body weight which happens when intake of calories exceeds expenditure of calories. In addition, there are numbers of 'obesogenic' factors, in other words, factors that promoting obesity which result in increased energy consumption and decreased energy expenditure namely declining level of physical labour, higher level of food consumption (high in fat content) and etc.[4]

In addition, dieting and physical exercise are the main solution for obesity. Balanced diet is helpful in decreasing or reducing consumption of energy rich foods such as those high in fat and sugars while increasing the intake of dietary fiber. Besides, anti-obesity drugs might be used to counter obesity by reducing the appetite or decreasing fat absorption when used together with a suitable diet.[3]

However, other than implementation of dieting, physical exercise or medication, active gaming seems to be an alternative yet effective solution. Active gaming or Exergaming is represented by video games that are also a form of exercise. Exergaming relies on utilization of body motion tracking technology. Such genre of games has been lauded for promoting an active lifestyle.[5]

With the introduction of Microsoft Kinect, active gaming has been seen as a great opportunity to develop active game with full body tracking feature which provides an engaging, immersive and interactive gameplay that could be pivotal in counter the widespread epidemic of obesity. Conventional solution of battling obesity such as dieting, physical exercise and medication might be effective, it is definitely not really appealing to obese patient as the discipline demanding yet repetitive nature of the tasks greatly drives down their motivation.

By using Kinect, a full featured serious game targeted for obesity patient could be achieved by integrating its invaluable marker-less motion tracking technology into the application. According to a study conducted among 2831 children aged one to twelve showed that video game was positively related to elevated weight status, yet those data were only accounted for girls aged nine to twelve who played moderate amounts of games. Some evidence even suggests that video games (passive gaming) induces or results in higher energy expenditure among children. Besides, some anecdotal evidence shows that interactive video games or active games that requires intense physical movement are contributing positive impact.[6]

In short, active game with utilization of Microsoft Kinect provides an engaging, interactive yet revolutionary solution in combating the widespread epidemic of obesity.

1.2 PROBLEM STATEMENT

Obesity is a serious medical condition that have garnered lot of attention lately in which kids, teenagers as well as adults have seen a galloping rise in term of their body weight. Such widespread epidemic has reached an alarming rate that demands immediate solution and countermeasure to contain this situation. While exergames are grabbing the headlines as the revolutionary solution in battling obesity, such games are not specifically dedicated to obesity patient, in other words, there is no proper obesity oriented serious game that caters obesity patients. Besides, patients are unable to make full use of such games due to their complexity and the need for a more tailored program or therapy.

Besides, in conjunction with the introduction of Microsoft Kinect sensor, active games has been afforded an opportunity for expansion while reaching out to more potential users which is beneficial towards combating obesity. However, there is problem lies in the developed games is that the gameplay offered simply could not engage users/ patients to be committed or motivated in carrying out the required task especially for obesity patient. Engaging gameplay represents a pivotal element to be kept in mind when developing an obesity oriented game as the game contents needed to be constantly motivate users in performing more exercises. While there is significant effort in commercial active games such as Kinect Fitness, Dance Central and etc; yet, there is still room for improvement for a full featured physical therapy game that could enhance the motivation level of targeted patients.

In addition, while physical based exercise represents a viable solution in battling obesity that should be definitely employed in an active game; yet, cognition based gameplay could not be left out. By combining both physical and cognition gameplays, users or patients could be further motivated or attracted in playing the game. Besides, a study suggested that passive gaming that requires users' cognitive skills is essentially a virtual exercise itself as it helps burn energy.[7] Thus, the combination of active gaming with cognition element could be an effective countermeasure against obesity.

As such, problem statements for this project are defined as below:

1. There is no tailor or custom made active games specifically dedicated for obesity patients
2. Lack of engaging elements or gameplays to motivate users in using the system.
3. Lack of cognition element in the existing active games that could be pivotal in motivating users.

1.3 OBJECTIVE

1. To develop a Microsoft Kinect based game to solve obesity problem by promoting active gaming.
2. To create an engaging gameplays to foster interaction between system and users in order to make users committed in performing the given tasks.
3. To develop gameplays that requires both physical as well as cognition interaction from users.

1.4 SCOPE

1. The system is targeted on users who are categorized as overweight or obese in term of their respective BMI measurement who aged between 10 and 30.

1.5 THESIS ORGANIZATION

There are six chapters in this thesis:

i. Chapter 1 – Introduction of the research on Microsoft Kinect in solving obesity problem
ii. Chapter 2 – Literature review on existing projects or approaches
iii. Chapter 3 – Methodology on implementation of the system
iv. Chapter 4 – Design of the respective system
v. Chapter 5 – Implementation of the system
vi. Chapter 6 – Result and discussion of the system
vii. Chapter 7 – Conclusion

CHAPTER 2: LITERATURE REVIEW

2.1 OBESITY

Obesity could be defined as a medical condition of whom body mass index (BMI), a formula or measurement obtained by dividing a person's weight by the square of person's height, is greater than $30kg/m^2$ [3]. Lot of efforts have been made to combat obesity, be it the conventional method of physical exercise or dieting; yet, there is a viable solution that could possibly revolutionize the approach in tackling obesity in which we called active game. In this context, serious game for obesity would be the representation of active game.

2.2 SERIOUS GAME

Serious game could be defined as a games designed with primary purpose other than pure entertainment. Serious game is not categorized under certain genres of games but represent a whole category of games with different purposes such as educational games, political games, or real world simulation.[6] In this case, the development of serious game would be targeting obesity patients in an attempt to tackling the obesity with utilization of active gaming concept.

2.3 ACTIVE GAMING

Active gaming or exergaming is a term that represents video games which are also a form of exercise. Exergaming utilizes technology that tracks body movement or reactions. Exergames could be seen as an alternative games that bring more fun as well as better immersion level while promoting healthy lifestyle.[13]

2.4 MICROSOFT KINECT SENSOR

Microsoft Kinect sensor codenamed Project Natal is a motion sensing input device by Microsoft for Xbox gaming consoles as well as Windows PC. Manufactured around a webcam-style add-on peripheral, Kinect enables users to control and interact with their console/computer without needing a game controller. In addition, Kinect enable interaction with PC and consoles through a natural user interface using gestures and spoken commands.[9] In this case, the development of serious game would utilize the invaluable marker-less full body motion tracking technology offered by Microsoft Kinect.

2.5 SERIOUS GAME WITH MICROSOFT KINECT SENSOR

Utilization of Microsoft Kinect in serious game has seen the expansion of capabilities in serious game while charting unfamiliar territory that is previously impossible in term of implementation. With Kinect, for example, physiotherapy at home without the presence of professional therapist could be made possible. The invaluable technology of marker-less motion sensing has given an opportunity to implement revolutionary gameplay without any hassle regarding hardware and cost. Furthermore, the technology is also being used in the development of serious game regarding rehabilitation, exercise as well as fitness. In this context, the development of serious game for obesity will utilize the motion tracking technology in tackling the obesity problem.

2.6 CONSTRAINTS OF CONSUMER FITNESS GAME

In solving the obesity problem, physical exercise has been made interactive and fun with the introduction of consumer fitness game, a game developed with active gaming in mind to promote active and healthy lifestyle among gamers. This solution has seen a positive impact in term of solving the obesity problem with patients seem committed in performing the given exercises. However, the major downfall of the consumer fitness game would be its gameplay mechanics; as fitness games ranging from dancing, sports mini games and etc are not exactly tailored to solve the obesity problem. Besides, the complexity nature of the gameplay such as dancing might put off the interest or commitment of obesity patients as the required moves or actions are too difficult to be done considering the physical capability of an obesity patient. In short, the consumer fitness games are not tailored made for obesity patients.

2.7 REVIEW OF EXISTING SYSTEM IN REGARDS OF SERIOUS GAMES WITH MICROSOFT KINECT

2.7.1 PRESCRIPTION SOFTWARE FOR RECOVERY AND REHABILITATION USING MICROSOFT KINECT

This system is designed to help in making rehabilitation tool a viable solution at home by using Microsoft Kinect.

In this system, in order enhance rehabilitation experiences of patient, the system or prescription software is divided into two parts, namely patient rehabilitation experiences (rehabilitation game), metrics and monitoring engines.

1. Patient Rehabilitation Experiences (PRE)

In PRE, a clinician is able to alter or configure the application to tailor the characteristics of their condition. The PRE could be defined as an interactive environment with objectives to be met by patients. By using gestures as input from Microsoft Kinect, patient is able to interact freely within the environment to perform certain set of tasks.

In addition, the PRE is designed with game mechanics in mind and the gameplay could be dynamically altered to fit the abilities of a patient in order to inspire or motivate patient to do better. If a patient finds it hard to complete the given action, then the next iteration of the required action would be simplified. The examples of PRE can be seen in Figure 2.1 below:

Figure 2.1 A PRE with a user deflecting the oncoming balls

2. Metrics for Healthcare Professionals

In this case, to make a new rehabilitation method to be deemed medically relevant, the method have to be proved to have a positive impact on patient recovery. Thus, the metrics represent an evidence that could indicate the effectiveness of the system.

The data stored could be used to monitor the progress over time of the patient and this helps in determine the effect of treatment quantitatively.[12]

2.7.2 ENHANCING EFFECTIVENESS OF MOTOR REHABILITATION USING KINECT MOTION SENSING TECHNOLOGY (KINECT-O-THERAPY)

This system is designed to enhance effectiveness of motor rehabilitation by utilizing motion sensing technology of Kinect.

In this system, in order to enhance motor rehabilitation, four motion sensing exercise routines in introduced in a game developed using Unity3D engine. Besides, the software or system has been designed in a way that interaction between user and system is seamless with the natural user interface offered by Microsoft Kinect. The system is made up of four mini games that are translated by conventional rehabilitation exercises that are advocated by professional physiotherapist. Besides, the performance data recorded in game could be recorded into database and can be viewed by both patient doctor online.

The four games introduced in this system are namely Shoulder Abduction, Balloon Pop, Path Follower and Play Along. These game can be shown in Figure 2.2 below:

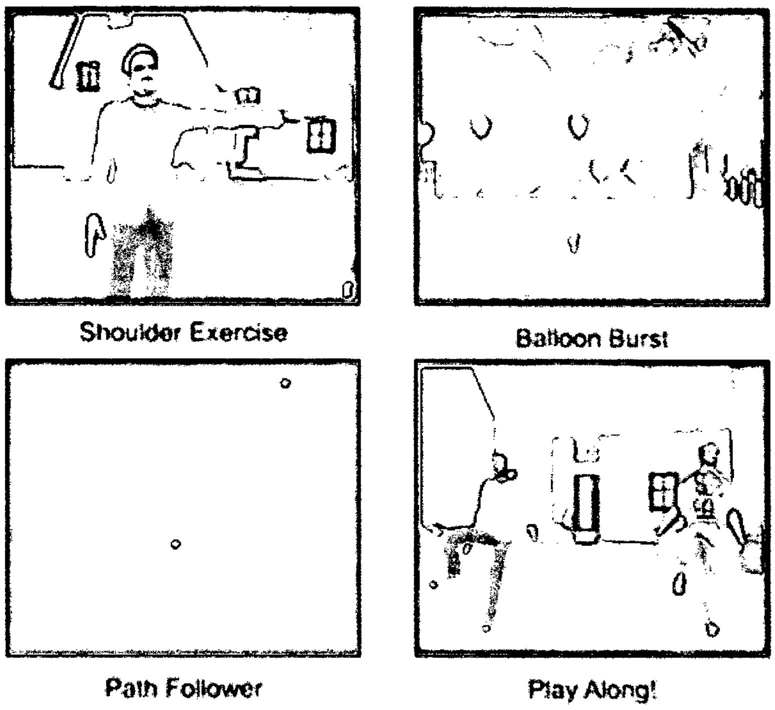


Figure 2.2 Four Mini Games in Kinect-O-Therapy

1. Shoulder Abduction

This routine has been integrated into the game to enhance the degree of shoulder abduction of a patient. The patient is required to move his left and right shoulder to make a straight line with his shoulder in order to register a successful attempt. If the player move his shoulder beyond the specified level, the attempt is considered invalid. In addition, if the incorrect movements is registered, an auditory clue is given to indicate correct and incorrect movements.

Balloon Pop

The exercise routine is included to helps the patient enhance his hand stability and hand-eye coordination in the X-Y plane. In this game, the patient is required to burst all the balloons as quickly as possible. The patient has to place his hand on the balloon and wait for two seconds to make the balloon pops.

2. Path Follower

This exercise routine enhances the balance and coordination while walking. In this game, the patient has to walk on the path indicated on the screen. As patient successfully steps in the correct path, the path turns green and if patient steps out of the path; the path turns red.

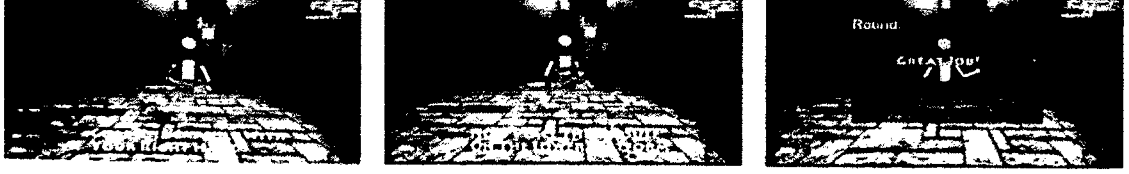
3. Play Along

This mini games allows the participation of two patients in a collaborative manner. The routine is effective when one of the participants is physically challenged while the other are not. In this game, the former or the patient will try to imitate his partner's action and would encourage himself to strive to the furthest extent possible.[11]

2.7.3 TOWARDS PERVASIVE PHYSICAL REHABILITATION USING MICROSOFT KINECT

This system is designed to demonstrate the capability of Microsoft Kinect as a viable solution in term of rehabilitation tool with a game based rehabilitation application developed using Unity3D game engine.

In this system, motor task is emphasized with the aim of improving the motor capabilities of the patients. To enhance motor capabilities, a mini game is designed for patients to perform External Rotation which is the main focus of the system. This can be seen in Figure 2.3 below:



(a) At the beginning of the game, the player is instructed to grab the capsule using right hand (b) The green color indicates the player's movement is correct (c) The player successfully finishes the movement

Figure 2.3 A sequence of screenshots of our rehabilitation game

During the game, the patient is told to use his/her hand to move the virtual object from one side to another side of the screen. If this exercise is performed correctly which represent correct movement pattern, the path would be green in colour. Conversely, if the exercise is not performed correctly, the path becomes red and arrows appear on the screen to guide the player patient back into the correct position.

Lastly, when the patient has achieved the required repetition of movement in a correct way, the virtual object will be released from the patient's hand and the patients would be encouraged to try with different virtual object and consequently perform the movement again.[10]

2.7.4 KINECT SPORTS

This game is a consumer game that is designed to promote active gaming among gamers by utilizing Kinect motion-sensing peripheral. This system or game consists of collection of six sports simulations and eight mini games. The six sports included are Bowling, Boxing, Track & Field, Table Tennis, Beach Volleyball and Football. In order to play these games, consumer or gamers are required to mimic the actions performed in real-life sports namely throwing javelin or kicking a football.

1. Sports Games in Kinect Sports

1.1 Bowling

In order to play bowling in Kinect Sports, players are instructed to reach their left or right to take up a ball before swinging their arm forwards to perform bowl. In this game, players are challenged to knock down as many pins as possible within a time limit. Figure 4 shows us the gameplay of bowling in Kinect Sports.

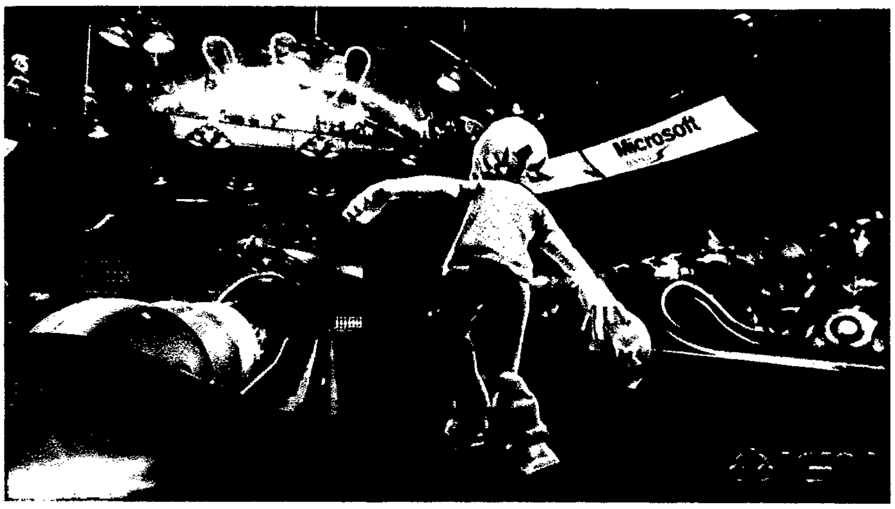


Figure 2.4 Bowling in Kinect Sports

1.2 Boxing

In this game, players are instructed to use their left and right arms to punch and block at both head and body height. Figure 2.5 shows us the gameplay inside the Boxing of Kinect Sports.

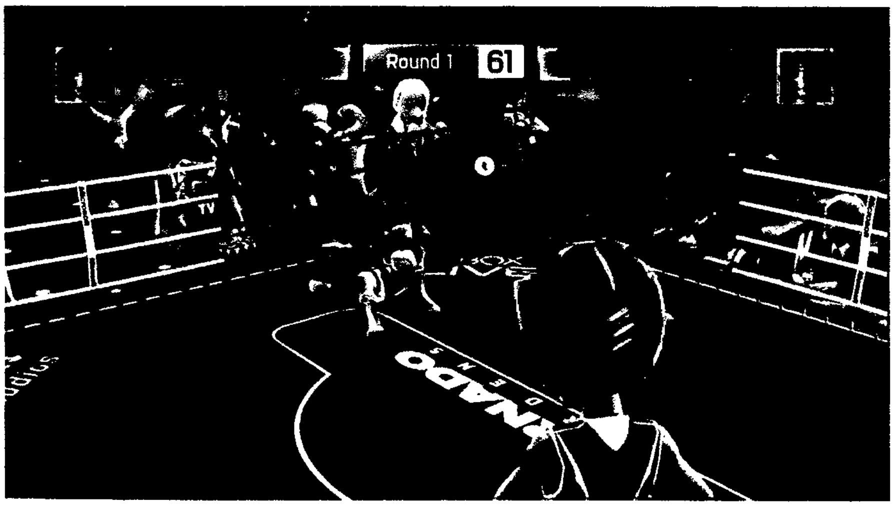


Figure 2.5 Boxing in Kinect Sports

1.3 Track & Field

In this game, the event is comprised of five separate events. The events are Sprint, Javelin, Long Jump, Discus and Hurdles. To play this game, players must be perform several actions namely player must jog on the spot to run, jump to clear hurdles, make a long jump and perform the relevant arm motion to throw a javelin or discus. Figure 6 shows us the main interface in the Track and Field game. In addition, Figure 2.7 and 2.8 shows us the mini games inside Track and Field namely Hurdles and Javelin.

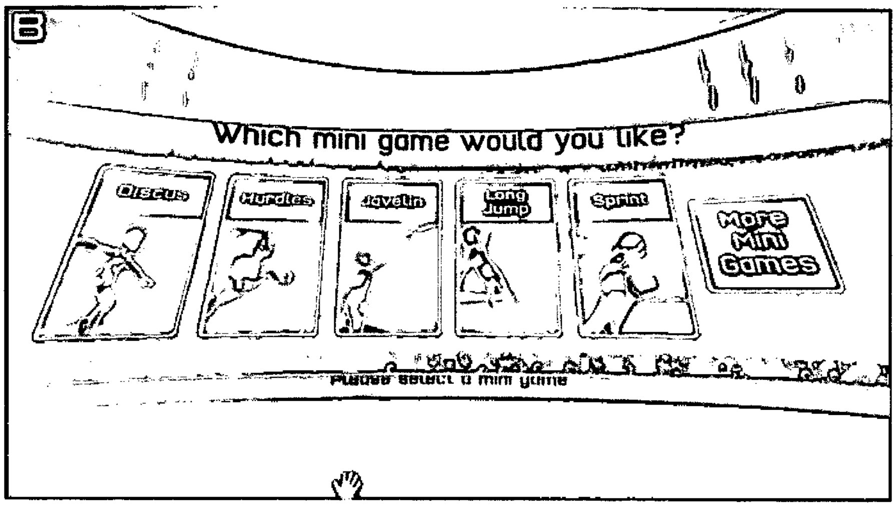


Figure 2.6 Track and Field

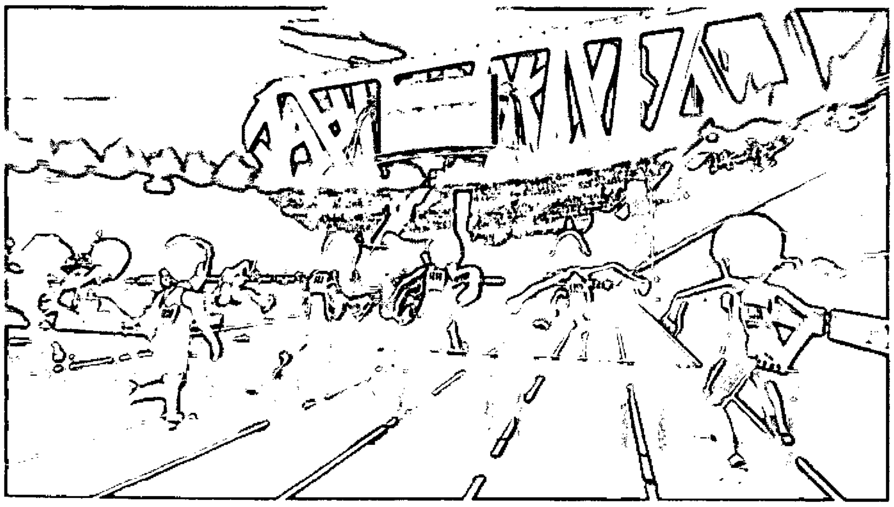


Figure 2.7 Hurdles (Track and Field)

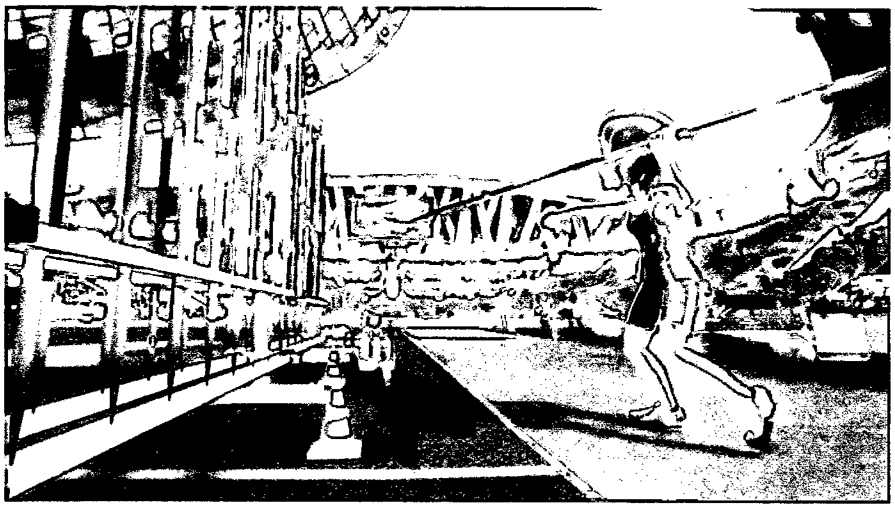


Figure 2.8 Javelin (Track and Field)

1.4 Table Tennis

In this game, players are instructed to reach their left or right to pick up a paddle before serving before performing topspin, backspin and smash shots. In this game, a player is challenged to maintain a single rally for as long as possible. Figure 2.9 shows us the gameplay of Table Tennis in Kinect Sports.

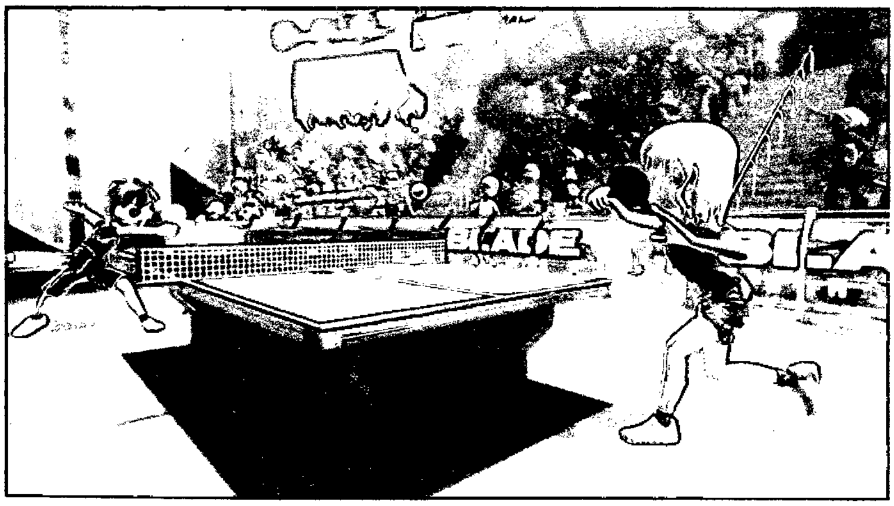


Figure 2.9 Table Tennis in Kinect Sports

1.5 Soccer

In this game, players are required to perform their actions in respective of their playing roles inside the gameplay. As an attacker, the player is instructed to kick the ball in order to pass or shoot. While playing as the defender, players is told to move from side to side to block passes and use their body to block shots. Figure 2.10 shows us the gameplay of Soccer inside Kinect Sports.

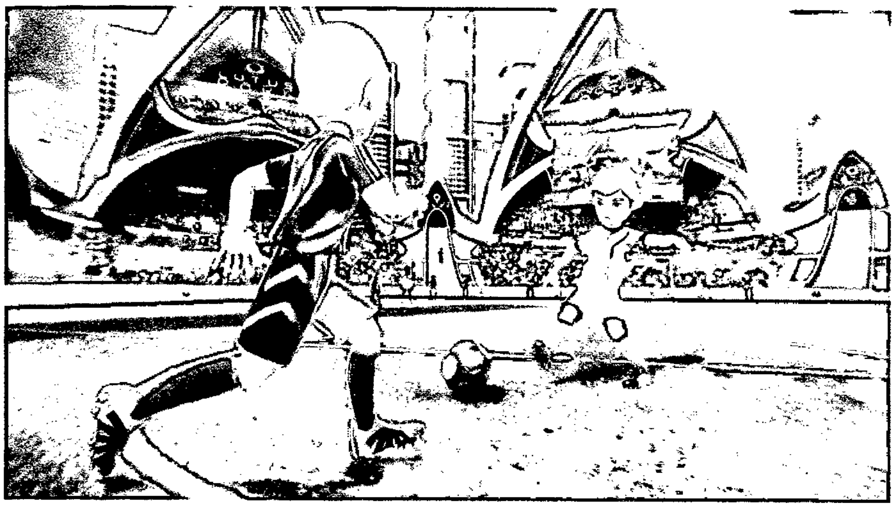


Figure 2.10 Soccer in Kinect Sports

1.6 Beach Volleyball

In this game, players serve by making an upward throwing motion with one hand and then a swinging motion with the other. Passing or return of the ball is done by bump, set and spike motions. Figure 2.11 shows us the gameplay if Beach Volleyball in Kinect Sports.[14]

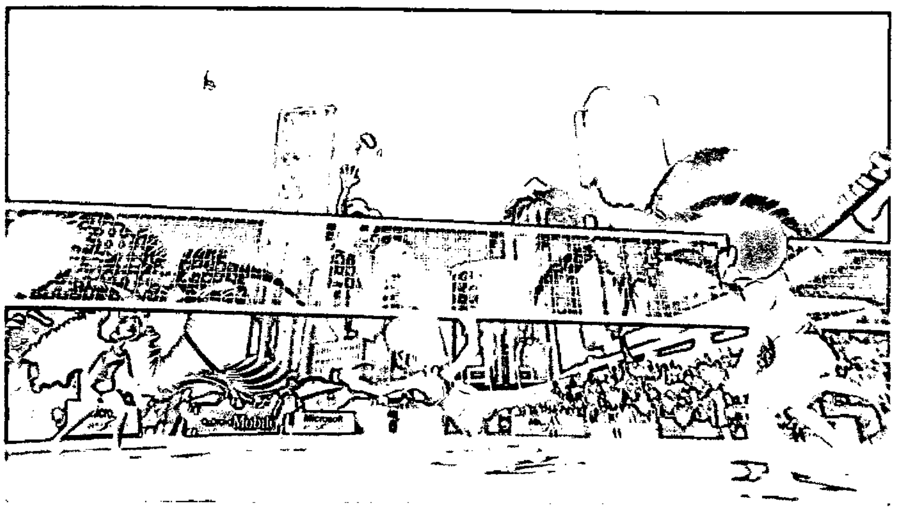


Figure 2.11 Beach Volleyball in Kinect Sports

In short, Kinect Sports is an active games that utilizes Kinect sensor motion tracking technology to incorporate marker-less tracking feature inside the game. However, the game mechanics and fitness regimen have to be redesigned to ensure the game is suitable for obesity patients considering their physicality attributes.

2.6.5 KINECT SPORTS RIVALS

This game is a consumer game that is designed to promote active gaming among gamers by utilizing Kinect motion-sensing peripheral. Kinect Sports Rivals represents another game in Kinect Sports series. In this release, mini games of sports genre such as bowling, wake racing, rock climbing, soccer, target shooting and tennis is included.

1. Sports Games in Kinect Sports Rivals

1.1 Wake Racing

In this game, player is required to dodge between the reclaimed hulls, launch yourself off ramps and impress the crowd with flips and performing tricks before going for a perfect landing while staying alert for floating mines. Figure 2.12 shows us the gameplay of Wake Racing in Kinects Sports Rivals.

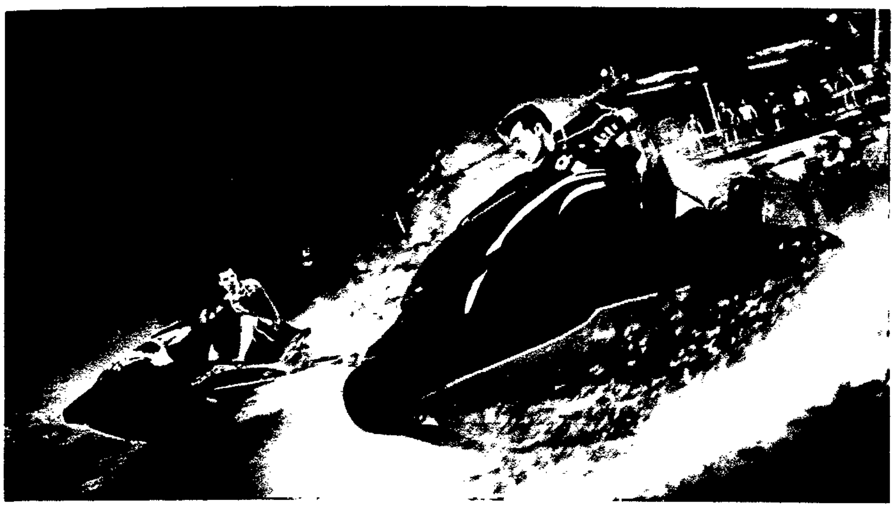


Figure 2.12 Wake Racing in Kinect Sports Rivals

1.2 Climbing

In this game, player is instructed to navigate through tricky overhang while keeping attention of their stamina level. Besides, player has to deal with gusty wind, hazardous handhold while climbing up to reach the summit. Figure 2.13 shows us the gameplay of Climbing in Kinects Sports Rivals.

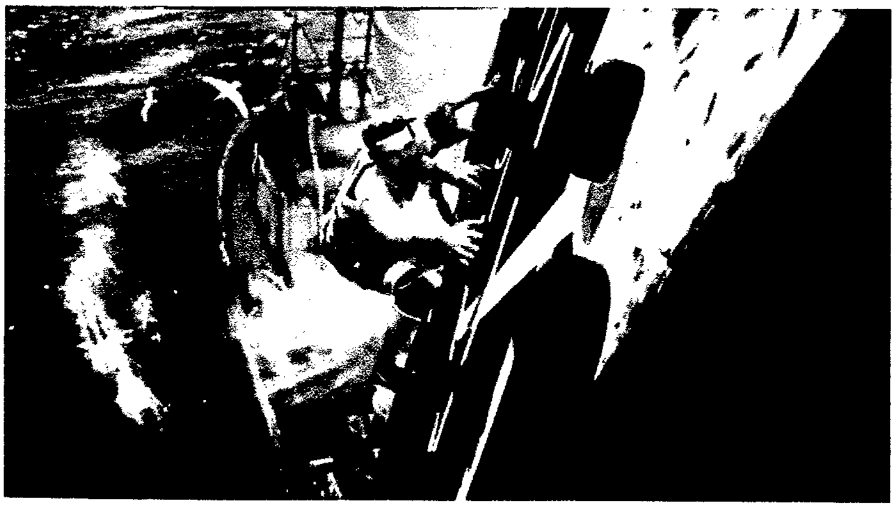


Figure 2.13 Climbing in Kinects Sports Rivals

1.3 Target Shooting

In this game, player is instructed to perform shooting to deal with waves of target that jumps, split and transform from moment to moment. Figure 2.14 shows us the gameplay of Target Shooting in Kinect Sports Rivals.

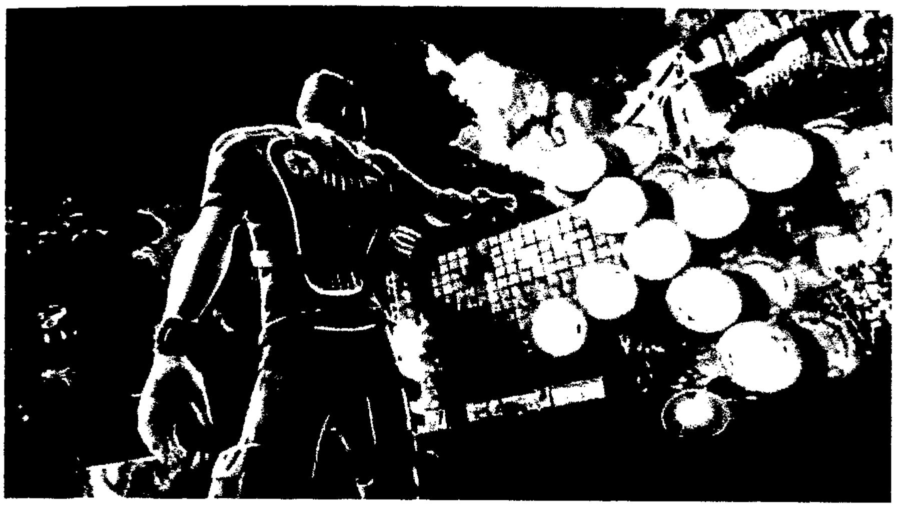


Figure 2.14 Target Shooting in Kinect Sports Rivals

1.4 Soccer

In this game, players are required to perform their actions in respective of their playing roles inside the gameplay. As an attacker, the player is instructed to kick the ball in order to pass or shoot. While playing as the defender, players is told to move from side to side to block passes and use their body to block shots. Figure 2.15 shows us the gameplay of Soccer in Kinect Sports Rivals.

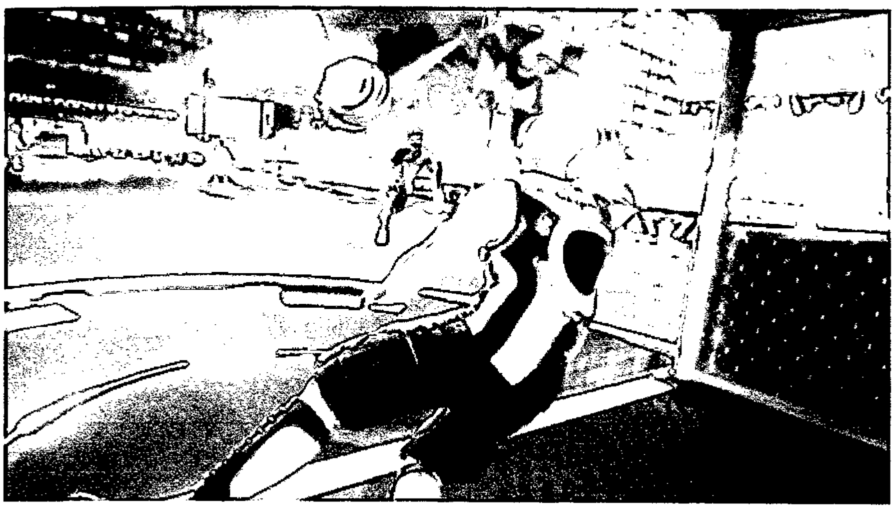


Figure 2.15 Soccer in Kinect Sports Rivals

1.5 Bowling

In this game, player is instructed to grab the bowling ball, line up for a shot and smash home a strike. Figure 2.15 shows us the gameplay of Bowling in Kinect Sports Rivals.

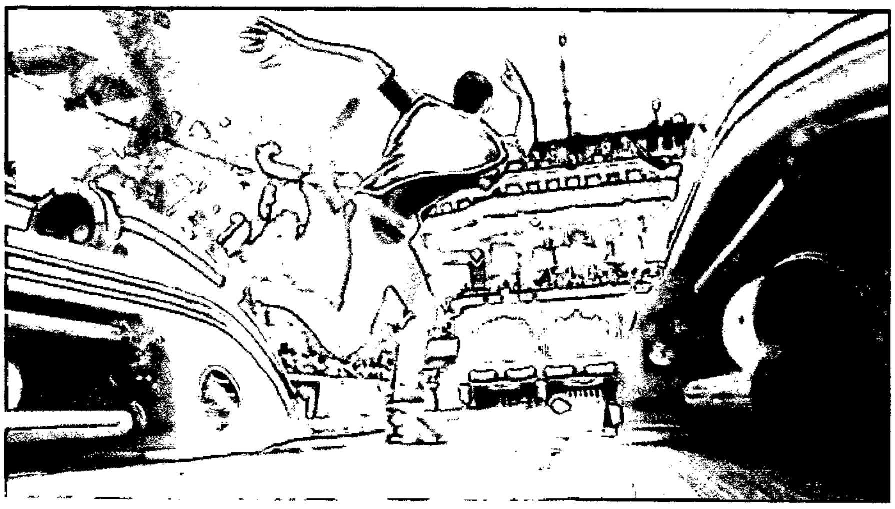


Figure 2.16 Bowling in Kinect Sports Rivals

1.6 Tennis

In this game, player is required to perform actions such as slice, lob and smash to obtain victory over rivals. Figure 2.17 shows us the gameplay of Tennis in Kinect Sports Rivals.[14]

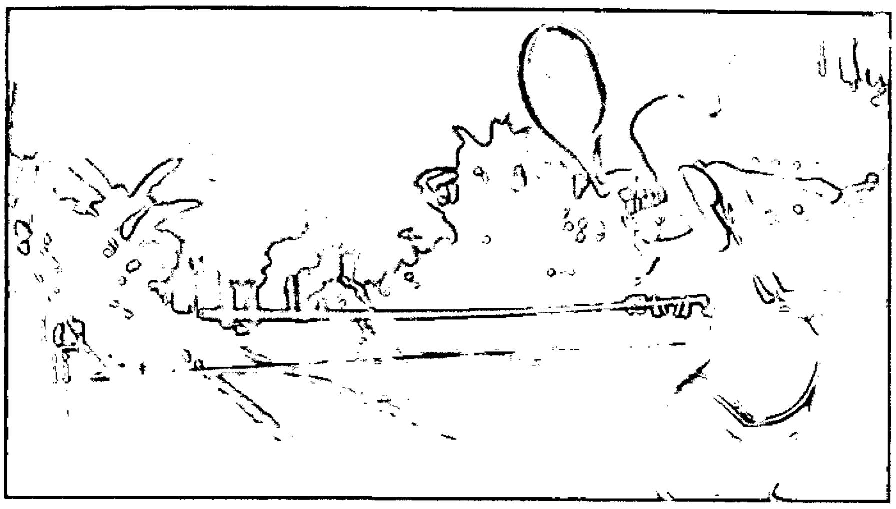


Figure 2.17 Tennis in Kinect Sports Rivals

In short, Kinect Sports Rivals represent a major upgrade over Kinect Sports with more interactive game mechanics offered. In addition, the release of this game represent a major boost in term of exergaming as to promote healthy lifestyle among gamers. However, being a consumer recreational sports game, Kinect Sports Rivals could not necessarily be the solution for obesity patient as fitness regimen and gameplay mechanics have to be redesigned to suit obesity patient.

2.8 LIMITATION OF EXISTING SYSTEMS

Table 2.1 Limitation of Existing System

	Kinect Sports	Kinect Sports Rivals	Towards Pervasive Physical Rehabilitation Using Microsoft Kinect	Proposed System
1. Active game for obesity patient	Kinect Sports is an active game that promotes or integrates exercises in a collection of mini games. However, the target consumer is not primarily obesity patient. In other words, exercises designed in this game is not corresponding	Kinect Sport Rivals represents an active game that promotes as well as integrates exercises in its gameplay with a collection of mini games. However, the target consumer is not primarily obesity patient as the game cater for recreational exercise	This system is designed to assist patient with motor disabilities by integrating rehabilitation exercises in the system. However, the target consumer is not obesity patient. While the system does integrate exercise in its gameplay, yet, the exercise is designated for motor disabled patient. Thus, the	The proposed system will be designed from ground ups to cater obesity patient with the inclusion of appropriate exercises considering their physical abilities.

	obesity patients.	purpose. Besides, the designed exercises do not reflect the needs of obesity patients.	designed exercise is not a representation of the needs of obesity patient	
2. Engaging gameplay	Kinect Sports provides an interactive yet engaging gameplay by including variety of sports games. Besides, to encourage consumer in playing this game, Kinect Sports has integrated innovative scoring mechanism that could be stored online for rivals or	Kinect Sports Rivals provides an interactive and engaging gameplay with a collection of mini games that comprised of variety of sports games. In addition, apart from its innovative gaming mechanism which requires consumer to perform actions like they would in real life, Kinect Sports	In this system, the gameplay is interactive. However, the gameplay offered is not engaging as the game is designed to cater patient with motor disabilities, yet, this could pose a serious issue to the targeted patient as their motivation level is affected. The gameplay is not designed for obesity patient as	The proposed system will be developed with a full featured gameplay that cater obesity patients considering their physical abilities with the integration of scoring system as to engage consumer in performing the exercises. Besides, the gameplay will be designed with interaction

		friends to review. However, the designed gameplay is targeted for general audience, in other words, the gameplay might well be quite challenging for obesity patient considering their physical attributes.	Rivals too include an online scoring system. However, the gameplay in not designed specifically for obesity patient as it is developed for recreational exercise purpose. Thus, the gameplay would pose some problem for obesity patients considering their physicality.	it cater patient with motor disabilities.	of system and user in mind. The gameplay will accept natural gestures from user as input that will be reflected in the virtual environment as to provide an interactive yet engaging gameplay experience.
3.	Gameplay mechanics in term of physical and cognition	Kinects Sports offers gameplay that demands both physical as well as	Kinects Sports Rivals offers gameplay that demands both physical as well as	In this system, the gameplay is purely on physical interaction	In the proposed system, there will be a gameplay mechanic that emphasizes

cognition interaction. In order to progress to higher levels or obtain better scores, apart from performing physical actions; player needs to have certain strategy to clear the in game obstacles.	cognition commitment from players. In order to get better in game results or high scores, player need to think of on how the clear the in game obstacles apart from performing physical actions.	between user and the system. There is no cognition features lie in this gameplay as the game is designated to cater motor disabled person. However, without cognition commitment, motivation level of patient would be greatly reduced as the repetitive nature of the task is boring.	both physical and cognition interaction between user and the system. Apart from performing physical action, user/ patient will need to think of performing the appropriate physical action to tackle the incoming in game obstacles.

2.9 LITERATURE REVIEW ON MOTION SENSING HARDWARE

In this section, comparison as well as analysis would be performed in order to choose the best appropriate hardware for the intended system. The hardware to be compared are Microsoft Kinect Sensor, Nintendo Wii Remote and Sony Playstation Move respectively.

2.9.1 MICROSOFT KINECT SENSOR

Microsoft Kinect Sensor is a motion sensing hardware codenamed Project Natal in its development. This motion sensing hardware features a marker-less motion sensing technology that enables user to interact with console or computer without the need of other input devices such as game controller. In addition, Microsoft Kinect Sensor signifies the concept of Natural User Interface, in other words, your body is the controller for console or computer through the translation of gestures, spoken commands into inputs.

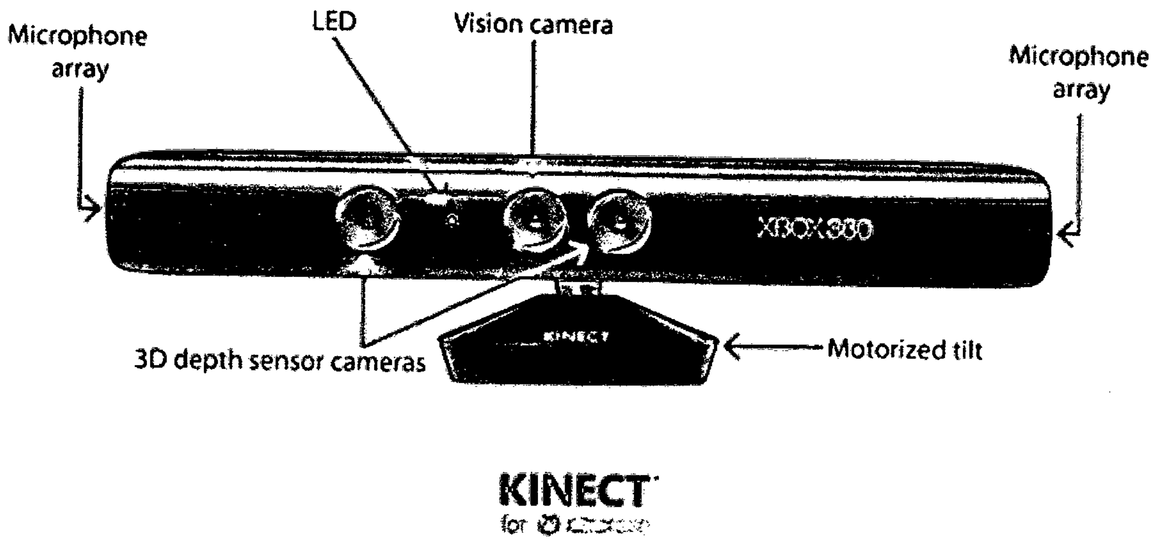


Figure 2.18 Microsoft Kinect Sensor with hardware labelling

Kinect is built on top of the software technology developed internally by Rare, a subsidiary of Microsoft Game Studios as well as a range camera technology by Israeli developer called Primesense which is credited for introducing a system that can interpret gestures command by using an infrared projector and a camera while using a special microchip to track the movement of the objects and individuals in 3D.

Moreover, Kinect is built similarly to a horizontal bar which connected to a small base with a motorized pivot. The hardware or device features an RGB camera, depth sensor and multiarray microphone. Such features of hardware provide us the capabilities of full body motion capture, facial recognition, as well as voice recognition.[15]

2.9.2 NINTENDO WII REMOTE

The Nintendo Wii Remote or simply Wiimote is the primary controller for Nintendo Wii console which is equipped with the main functionality of motion sensing which allows user to interact with and control or manipulate items in screen via gesture recognition and pointing via the utilization of accelerometer and optical sensor technology.

The Wii Remote utilizes a one-handed remote control-based design instead of conventional gaming controller. This design is intended to make motion sensitivity more intuitive or ergonomic as the design is also fitted for pointing while making it more appealing to broader consumer including non-gamers.

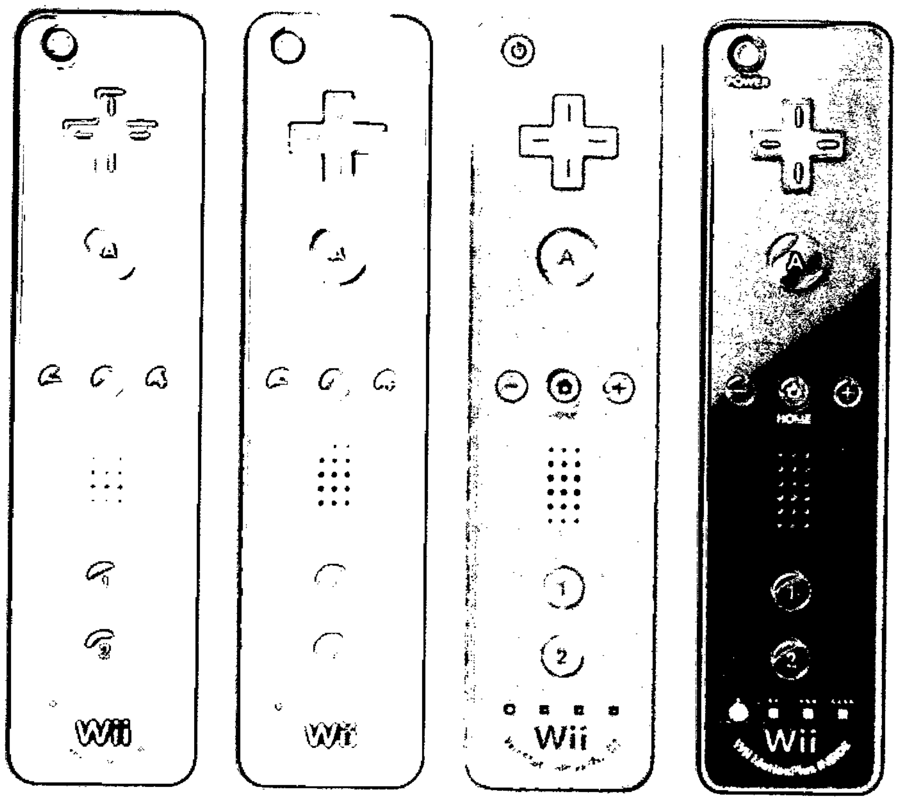


Figure 2.19 Nintendo Wii Remote

Moreover, the Wii Remote operates its main motion sensing functionality by utilizing the acceleration along three axes through an ADXL330 accelerometer. Besides, the remote also equipped with a PixArt optical sensor which in turn allows it to determine where the Wii Remote is pointing.[16]

2.9.3 SONY PLAYSTATION MOVE

Sony Playstation Move is a motion sensing game controller which is designed based around a handheld motion controller wand. The device uses the inertial sensors in the wand to detct its motion while the wand's position is tracked using a Playstation Webcam.

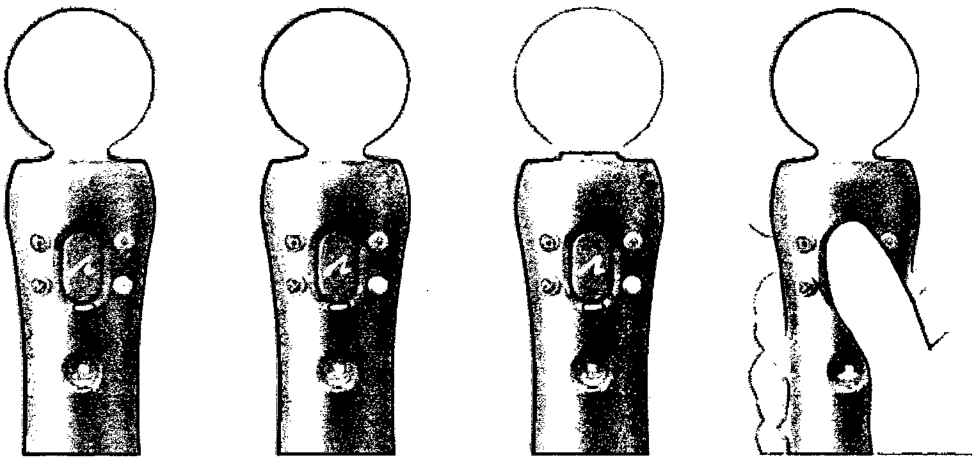


Figure 2.20 Playstation Move Controller

Furthermore, the main component of Playstation Move is a wand controller which allows user to interact with the console through motion as well as position in front of a Playstation Camera. The device features an orb at the head of wand which can glow in any full range of colors using RGB LEDs. Based on the color of the surround environment, the system can dynamically assign a color to the orb in order for it to be distinguished from the rest of the scene. The colored light serves as the active marker which offers tracking of position along the image plane by the camera. In addition, the size of the ball also allows system to simply determine the controller distance from the camera and thus allowing the motion to be tracked in 3D.

Furthermore, apart from a pair of inertial sensor, the Move is further equipped with a three axis linear accelerometer and a three axis angular rate sensor which is used to track rotation as well as overall sensor. Moreover, an internal magnetometer is also used to calibrate the orientation against the earth magnetic field in order to help correct the cumulative error by the drift sensor. [17]

2.9.4 COMPARISON OF MOTION SENSING HARDWARE

Table 2.2 Comparison chart of Move VS Wii VS Kinect[18]

Motion Controller	Playstation Move	Nintendo Wii	Microsoft Kinect
Motion Detection	Yes - 6-axis sensors built-in - XYZ detection by camera	Yes - 6-axis senors(with the addition of Wii MotionPlus) - XY detection by IR sensor	Yes - Controller-free - Sensors track body movement(skeletal traking)

In conclusion, the hardware chosen would be Microsoft Kinect as it provides a marker less motion sensing capability with the essential feature of full body movement or skeletal tracking that proves to be pivotal in this system or active game.

CHAPTER 3: METHODOLOGY

3.1 INTRODUCTION

The research was done on how to solve the rising obesity problem by integrating the marker-less motion tracking technology of Microsoft Kinect into active game as to tackle the problem by using a new approach. In addition, the research is aimed to produce an active game with Microsoft Kinect capabilities as to transform the ordinary living room into a workout arena for those obesity patients in order to provide a new form of workout in form of entertainment to keep patients stay motivated and committed to complete their workout regime. The project was done by doing researching on the solution of obesity and integrating it in the form of active gaming. Besides, the research was also done on how to motivate users to keep them stay committed to complete their workout regime by introducing an active game with entertainment values as to make users enjoy the gameplay while indirectly performing the workout regime.

The activities to consider during research development are finding solution of obesity and implement it in the active game, determining the multimedia element to be present in game as to keep users focus on completing the workout regime, finding solution to integrate Microsoft Kinect tracking technology into the game, determining the game engine to be used in order to develop the active game and lastly devising gameplays, game related functions (score, life, input from Microsoft Kinect Sensor) in the selected game engine.

3.2 METHODOLOGY

3.2.1 METHODOLOGY USED IN RESEARCH

The methodology that is used in this research could be represented by SDLC Waterfall Model. The Waterfall model could be defined as a linear sequential life cycle model. In this model, each phase must be completed before the next phase can begin and thus no overlapping in the phases.

The methodology of Waterfall model can be illustrated in phases. This could be further clarified as below:

1. Requirement Analysis

In this phase, the research has been mainly done to collect information regarding problem that to be addressed in this research which is the widespread epidemic of obesity. In this case, research has been done to get in depth information regarding obesity especially its causes backed up by statistics for verification and the most importantly, the solution to counter this widespread epidemic that is shaping the generation in a very adverse way. Besides, efforts were being made to study on how to propose a solution to counter obesity via electronic means that could make obesity patient committed to do it. In addition, works has been done to choose the best suitable electronic hardware, in this case, Microsoft Kinect Sensor for the proposed system design, in other words, an active game that is developed to cater obesity patient.

2. System Design

In this phase, design of system is being crafted as to cater the requirements into the gameplays of the active game for obesity patient. In addition, consideration such that actions or precisely, gestures that could be executed by obesity patient and not to stress them into executing any other extreme moves that could potentially lead to injury.

Besides, a gameplay mechanic is designed to make use of the proposed gestures that prove to be beneficial to obesity patient in order to provide an engaging gaming environment that catered specifically for them. Speaking of gestures that are designed to be executed by the user, running gesture or more precisely, brisk walking or jogging gesture is the main action needed to be executed by user in order to have proper cardiac workout. This is because cardiac workout represents the most effective way of dealing with obesity apart from a proper diet. Moreover, simple gestures such as raising their hands, leaning left to right and lastly clapping would complete the gesture lists. Such gestures is designed with consideration on user physical properties.

In addition to the game mechanics, the active game will be designed as corresponding to those infinite runner game in which user will have to brisk walk or jog to their own limit with engaging gameplay such as obstacles, health increment, enemies to keep user engaged in doing the exercises.

3. Implementation

In this phase, efforts are made to translate the design into an executable active game that is capable to work alongside with the chosen hardware or Microsoft Kinect Sensor. In addition, the designed game mechanics are translated into system by using the game engine of Unity3D, the official Kinect SDK and an interface called Kinect Wrapper to interfaces the functions available in Kinect SDK to be used in Unity3D. The Kinect Wrapper is downloadable script available on Unity Asset Store by Rumen Filkov.

Furthermore, works has been done to furnish those gameplays mechanic with visual elements such as 3D models, lighting and particle effects as well as brushing up the visual appeals with lens flares and lastly, animations.

Lastly, effort is made to develop the algorithm to recognize the gestures produced by user ranging from running, leaning left and right, raising left and right hands to clapping. In addition, much effort is done to complete the whole gameplays ranging from spawning of game objects, basic AI creation, enemy creation, animation blending using Mecanim to graphical user interface design to create an infinite runner game while using Microsoft Kinect Sensor.

4. Testing

In this phase, the research as well as the system would be verified its relevance and correctness by my project supervisor. In this testing, test cases will be assigned to my supervisor in order to complete the testing especially on the correctness of the system as according to the objectives. In addition, testing will be done on checking whether input or gestures in this case are received and well translated into the system. Besides, testing will also be done to ensure the system works as predicted in order to calculate the distance and calories achieved by the user.

While the effectiveness of the system could not be seen in a short time as obesity cannot be reduced in a short span of time. Thus, testing would only be possibly conducted to verify the system correctness as according to the requirements as well as objectives.

5. Deployment

In this phase, deployment of the system will be carried out on personal computer with USB port, Kinect SDK installed and lastly, the standalone Microsoft Kinect Sensor.

The deployment is done by uploading the system to internet cloud drive to make it accessible to for those that fulfill the basic requirements as above. Such deployment is to enhance the distribution of system to those needed rather than relying on physical media such as DVD.

6. Maintenance

In this phase, the maintenance will be carried out to improve the qualities of gesture tracking as well as overall gameplay to make it more engaging as well as exciting not to mention to make it more visually appealing as to reach broader audience. In addition, system could be upgraded to make it more efficient in term of memory and CPU as well as GPU usage so that the system could be accessible by personal computer with basic hardware specification.

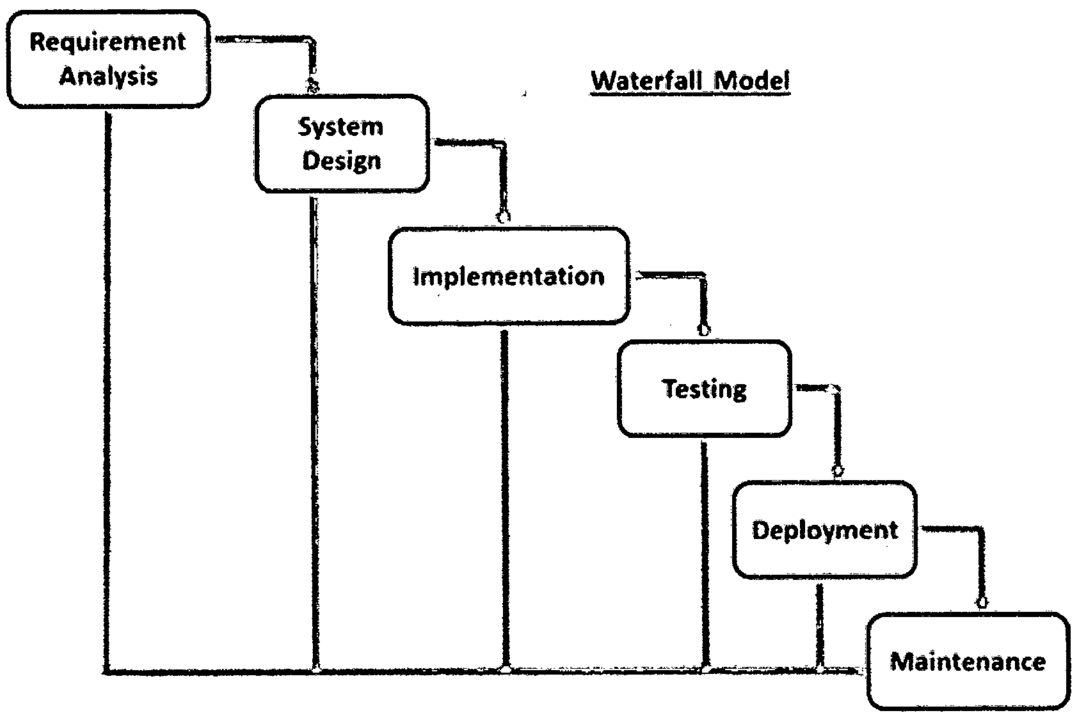


Figure 3.1 SDLC Waterfall Model[19]

3.2.2 FLOW OF SYSTEM

In order to implement the obesity solution inside the designated active game, the gameplay has been designed to include the essential cardiac workout, in this case, brisk walking or jogging movement from users as to consider their physical capabilities because running represents a huge challenge for obesity patients. In addition, the effort was made in finding solution to integrate Microsoft Kinect motion tracking ability inside the targeted game engine which is Unity3d in this case.

Moreover, the element of inside the gameplay has been designed to provide an interactive experience to users such as the introduction of obstacles in user's path, coins to be collected in collecting score. All this is done by implementing the natural user interface provided in Microsoft Kinect as playing the game requires real-time movement of user as input in order to interact within game environment. In addition, the progress or score in the game is displayed or identified by calculating the calories burned by the users while performing the workout, in other words, user is able to know an approximate count of calories burned in game.

1. Game Design

The character has to run through a specific level in order to complete the game. The level would be categorized based on their difficulty level as harder level requires more cardiac workout from user, in other words, the training regime would be harder and more intense. Upon successful attempt in completing a level, user would be directed into a more difficult level. Inside the gameplay, there will be obstacles which is blocking the user's path and in order to bypass the obstacles, the user would have to evade or block it in order to preserve his/ her health as each impact from the hitting will reduce a live of the character with a maximum of three lives. Besides, there will be coins in which user have to reach out with their arms while jogging or brisk walking through the scene. Lastly, the score would be displayed as calories burned of user, distance that the user has run as well as time used to complete the workout.

2. Game function

2.1 Input

The input function is used to get the input from user through the Microsoft Kinect sensor in the form of skeletal joint coordinates and translate it into the joint of game character in local space and lastly enable the movement of the character or avatar in game.

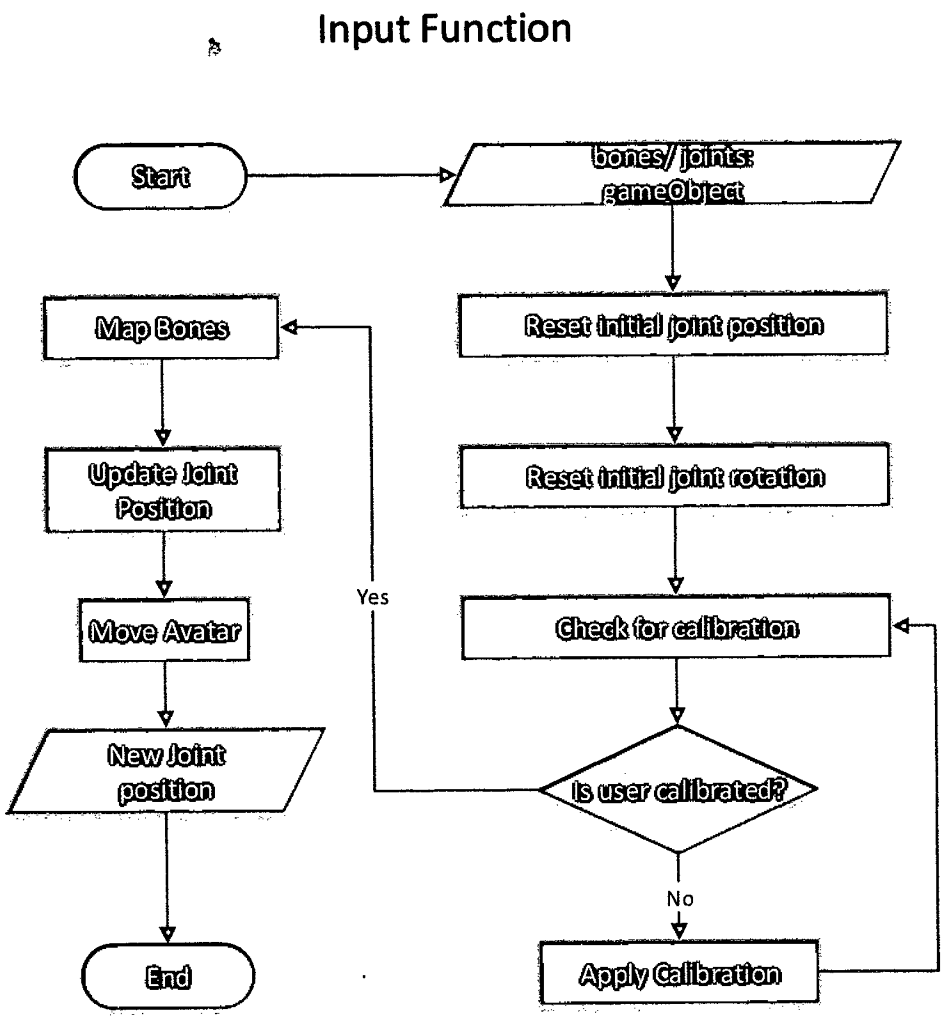


Figure 3.2 Flowchart of Input Function

2.2 Health and Kill

The health and kill function is used to identify the collision of game character with the obstacles and thus reduce its lives. After its lives have been reduced to zero, the game character will be destroyed which signifies the end of the gameplay.

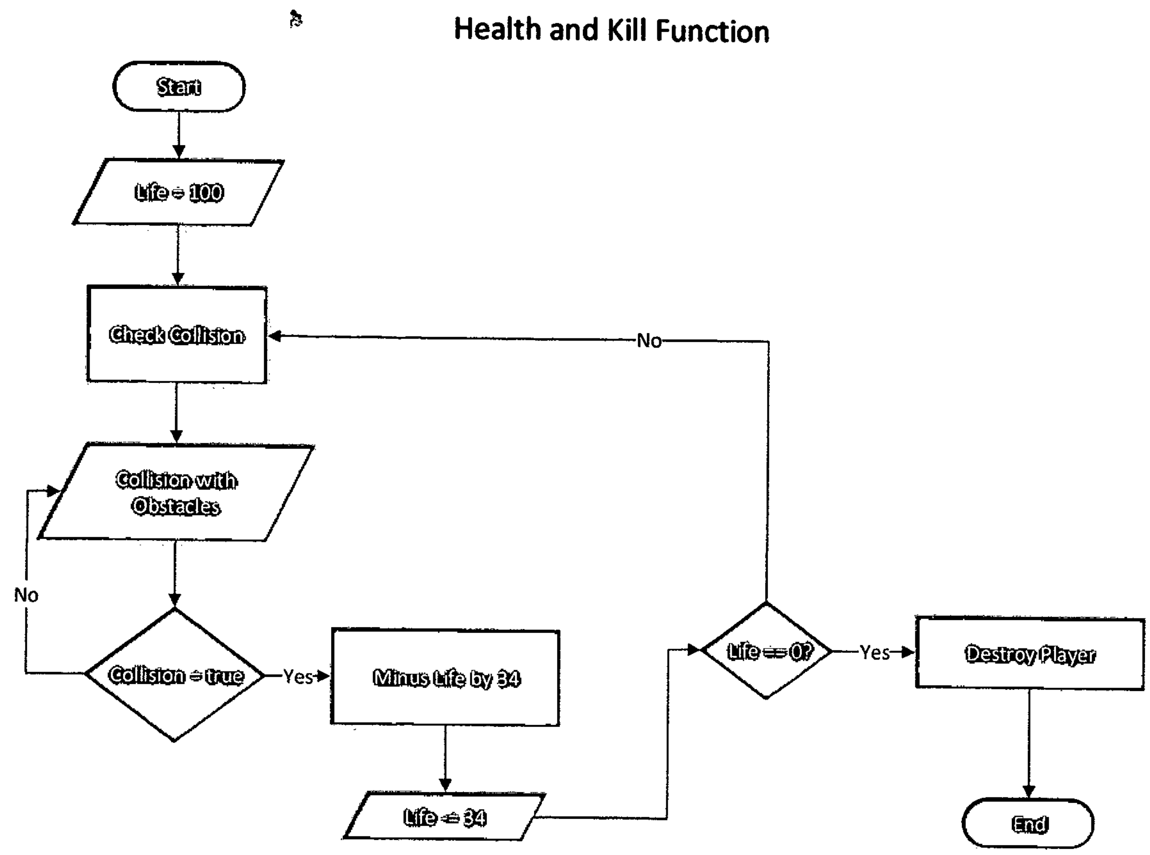


Figure 3.3 Flowchart of Health and Kill Function

2.3 Score

The score function is used to calculate the game score, in other words, calories and distance achieved by the user. Firstly, the function starts off by tracking the distance that the user has covered and afterwards, calculate the amount of calories burned.

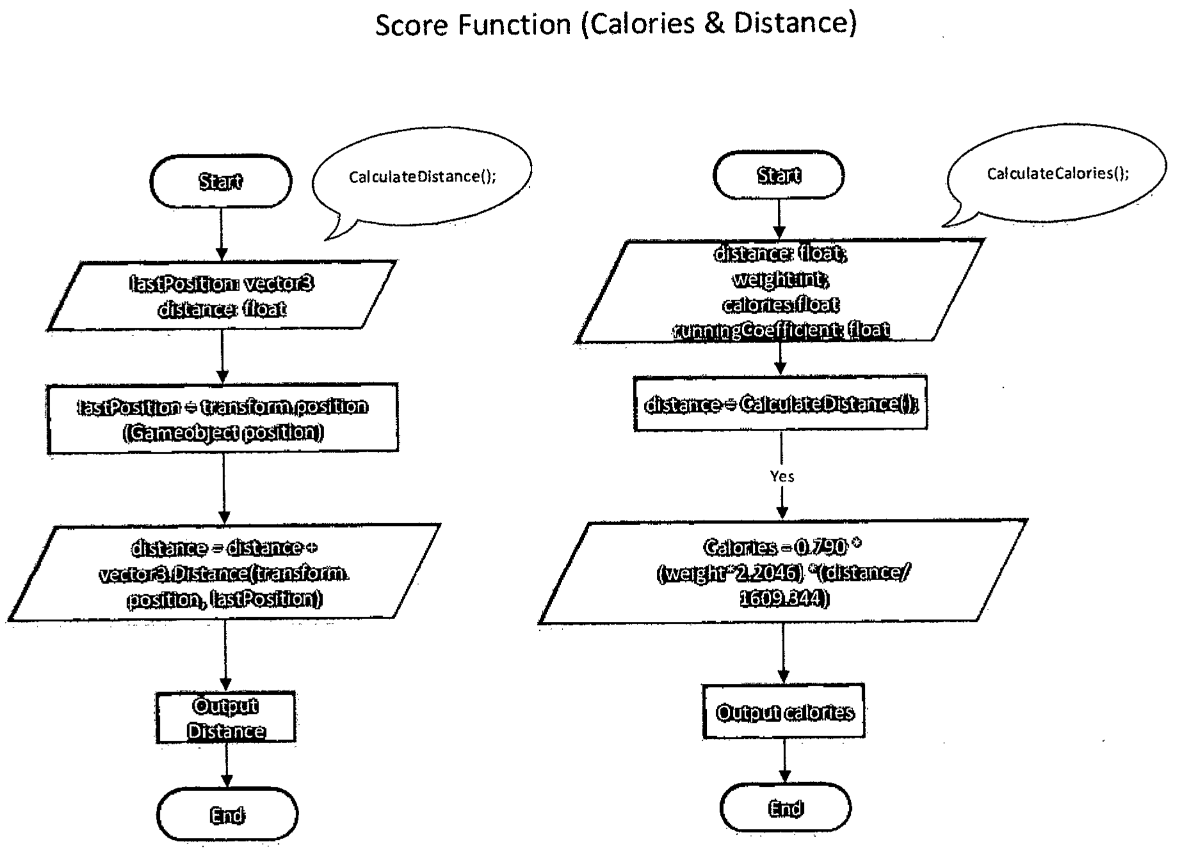


Figure 3.4 Flowchart of Score Function

3. Storyboard

The image shown below represents the scene inside the gameplay with the respective game elements which are explained below:

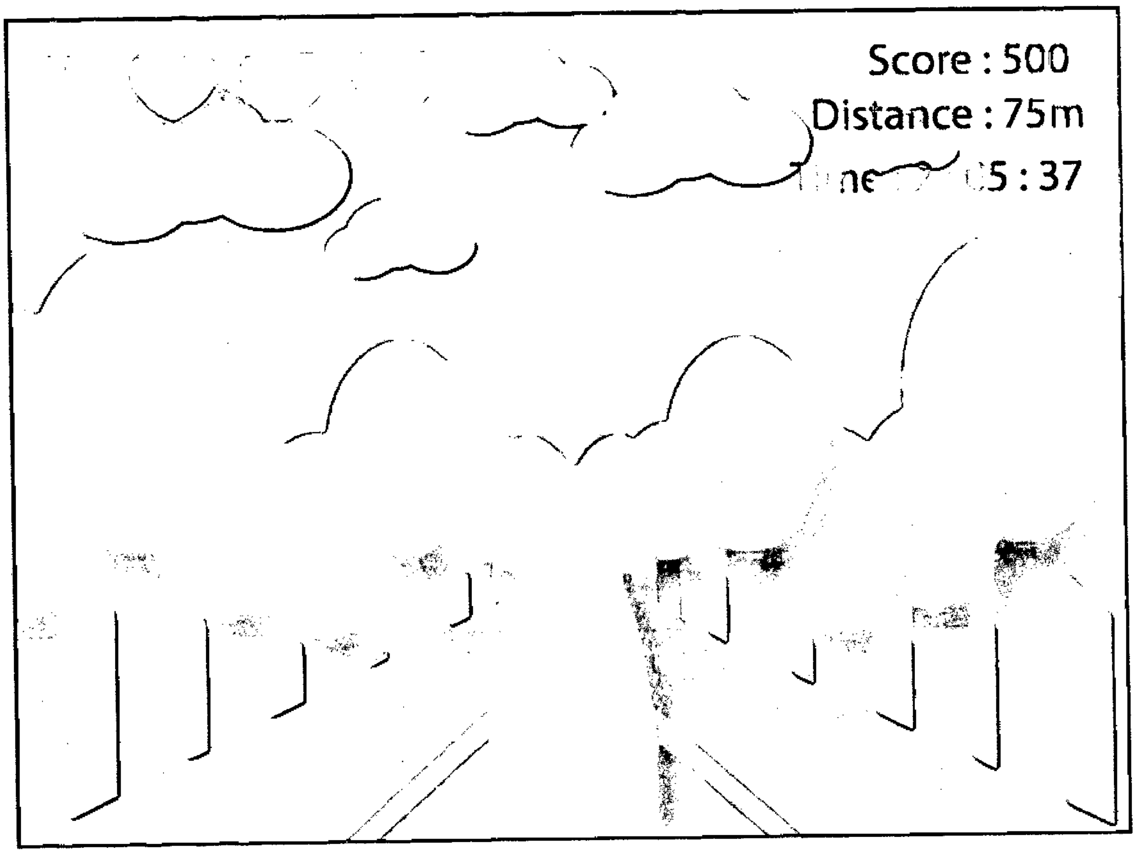


Figure 3.5 Storyboard

3.1 Character

The game character or avatar is used to receive the input from user through Microsoft Kinect and translate the input into its movement in game.

3.2 Obstacles

The obstacles is designed to reduce player's health as to make sure the gameplay is interactive and engaging to motivate user to complete their workout regime.

3.3 Calories burned

The calories burned is calculated by using formula from http://www.coolrunning.com/engine/4/4_1/94.shtml website.[20]

Calories = 0.790 * (weight * 2.2046) * (distance / 1609.344)

3.4 Distance

The distance is the value of workout distance that user has run or jogged through by converting the cover distance in game into unit in meter.

4. Hardware and Software

The hardware needed in making this research is a Unity3d compatible computer as well as the Microsoft Kinect sensor as the input device in the development of the active game. Regarding the software, the required software is Unity3d in order to develop the active game as well as Kinect SDK as to provide functionalities such as skeletal tracking algorithm to be ported into Unity3d.

CHAPTER 4: DESIGN AND IMPLEMENTATION

This chapter will describe the entire process in this development of project with clarification on what is needed to be done in order to complete the whole project.

4.1 DESIGN OF SYSTEM MODEL/ARCHITECTURE

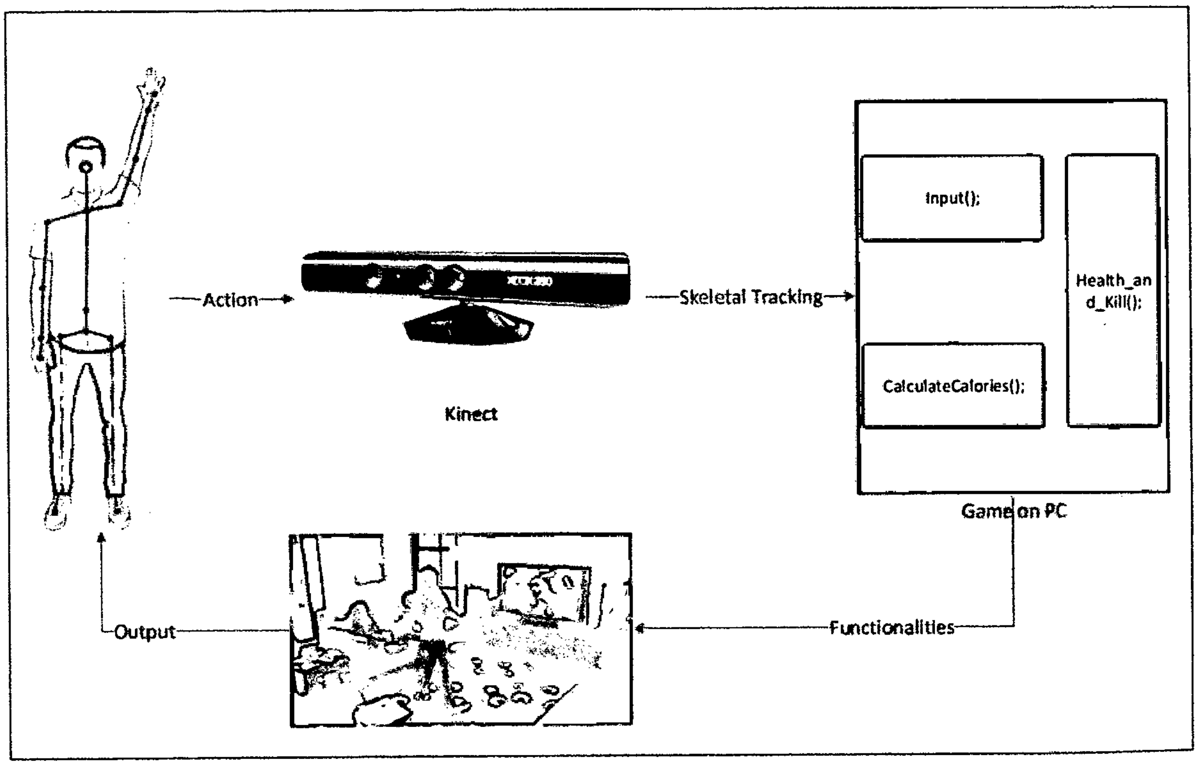


Figure 4.1 Design of system model

The model shown above illustrates the processes as well as data flows that is required in this system.

Firstly, the action is executed by the user or more precisely, obesity patient in this case; afterwards, Kinect sensor will receive the action or data in form of RGB and depth images which would be converted through algorithm available inside Kinect SDK into joints positions in 3D space with XYZ coordinates information. Next, this data would then be transported into the system or game by a script called Kinect Wrapper developed by Rumen Filkov(downloadable from Unity Assets Store) which interfaces the Kinect SDK functionalities of getting the joints positions to be computed and applied according to the algorithm in gestures tracking.

For example, in order to track the running gesture from user, the algorithm would be crafted to make use of the knees joint positions to determine whether the gesture is executed or not. In the case of running gesture, knee positions would be detected their heights, a cycle of walk or run would be completed if one knee is higher than the other knee. However, the algorithm is also designed to detect alternating knee motion which mean user would need to alternate the lifting knee motion and not just keep lifting one single knee.

After the gesture is computed and recognize by the system, the animation of the in game avatar would be executed. This motion will be reflected by the movement of avatar in game and thus, complete the cycle of the processes as well as data flows.

4.2 IMPLEMENTATION OF SYSTEM

4.2.1 STRUCTURE OF RESEARCH ON THE SYSTEM

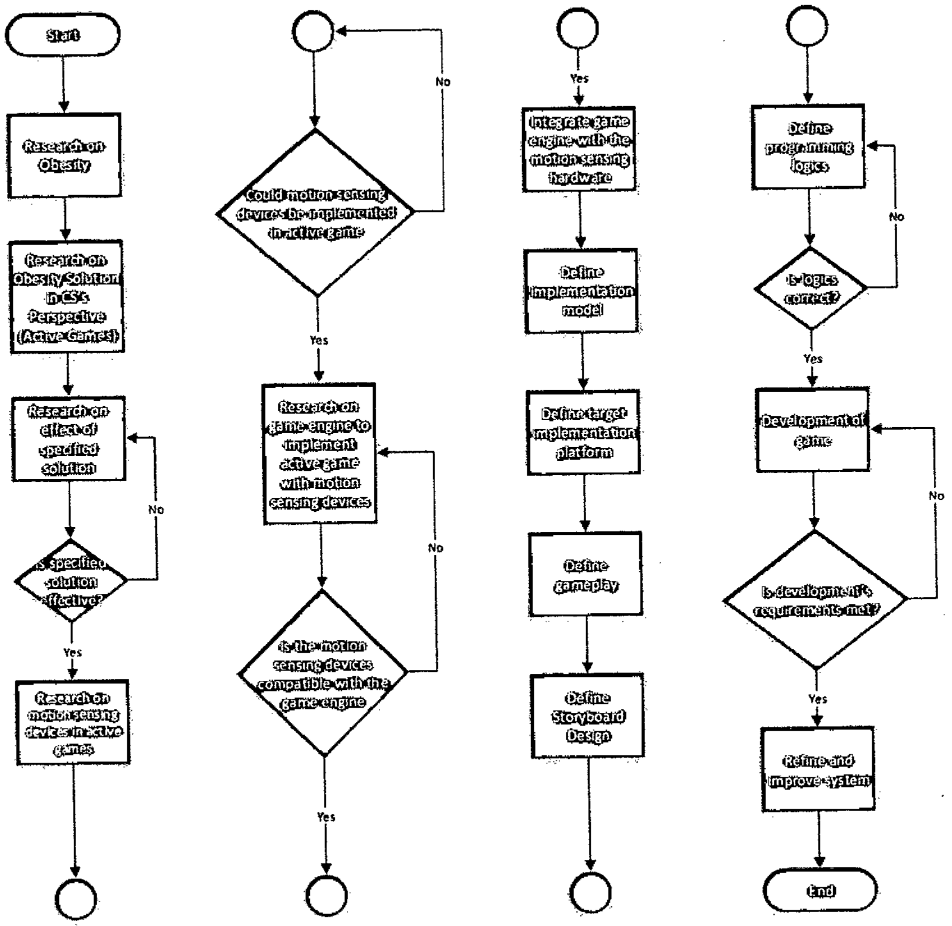

Start
Research on Obesity
Research on Obesity Solution in CS's Perspective (Active Games)
Research on effect of specified solution
Is specified solution effective?
No
Yes
Research on motion sensing devices in active games
Could motion sensing devices be implemented in active game
No
Yes
Research on game engine to implement active game with motion sensing devices
Is the motion sensing devices compatible with the game engine
No
Yes
Yes
Integrate game engine with the motion sensing hardware
Define implementation model
Define target implementation platform
Define gameplay
Define Storyboard Design
Define programming logics
Is logics correct?
No
Yes
Development of game
Is development's requirements met?
No
Yes
Refine and improve system
End

Figure 4.2 Flowchart of Research Flow

4.2.2 ACQUIRING JOINTS COORDINATES

Firstly, in order to establish communication between Unity3D and Kinect Sensor or more precisely Kinect SDK, an interface called Kinect Wrapper is used to access the libraries of Kinect SDK via kinect10.dll. The Kinect Wrapper is a plugin or codes written in C# in this case to interface the functionalities available in Kinect SDK such as acquiring depth image, skeleton lines and etc from the sensor. In this project, the Kinect Wrapper is a free download from Unity3D Assets Store by Rumen Filkov in form of an Unity Package named Kinect with MS SDK.[13]

In order to track the joint coordinates of user in Unity3D which comes in meters, function that interfaces the Kinect SDK via Kinect Wrapper need to be introduced and in this case, an essential script called Kinect Manager is introduced by the author or Rumen Filkov. This script acts as the main playmaker in the following scripts that will be covered in this chapter. The Kinect Manager serves as the bridge as well as the main script that provides essential functions such as get depth images, RGB images, skeleton lines to enable the Kinect sensor could be accessed. Besides, with Kinect Manager, by using C# scripting in Unity3D, we could create an instance of a Kinect Manager in other script in order to access the public method/ functions or variables in the Kinect Manager.

In this case, the essential function that could be accessed by us in other scripts is the function on getting joint positions in form of coordinates measured in 3D space as meters. With the acquisition of those data, gestures could be programmed in order to provide user a natural user interface or ways to interact with this system or game with only just our body.

4.2.3 TRACKING OF GESTURES

1. Running

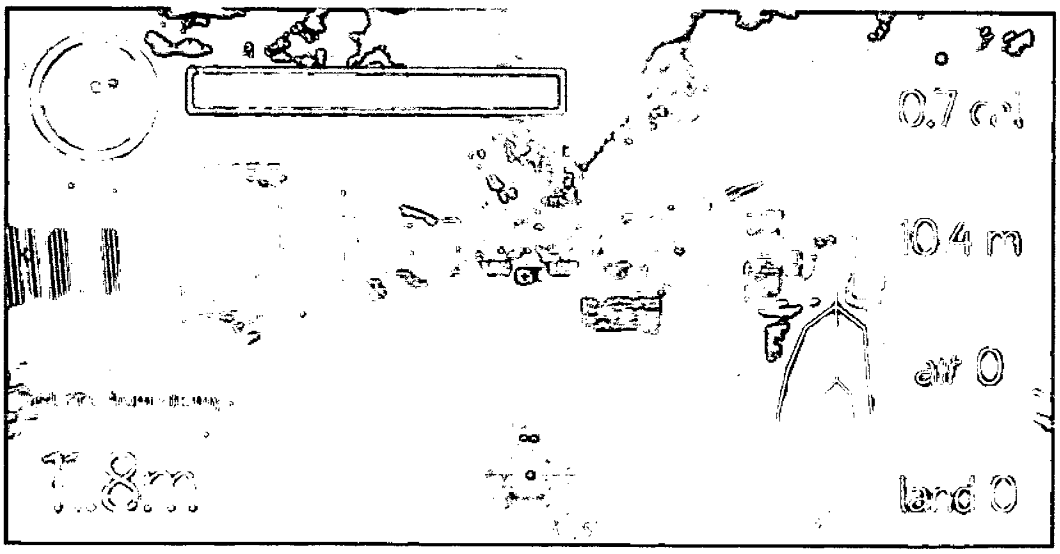


Figure 4.3 In-game avatar perform running animation when running gesture is detected

In this game, running represents the most important exercise that is integrated in this game, in fact, it is the core feature in this game in order to encourage cardiac workout from our target users. In order to track the gesture produced by users, conditions or simply steps have to be introduced into scripting of this gestures. This could be clarified by tracking the movement of running which could be translated into programming as tracking of users' knees positions. In order to enable the recognition of such gesture, computation will be done on users' respective knee position. In this context, the height or y position of knees positions would be compared to ensure walking or jogging steps are executed by users.

After tracking the knees' movements of users, a function or logics would be introduced to compute the speed in order to determine how fast the game avatar should go corresponding to your walking or jogging speed as to blend the animation between jogging and running using the powerful Mecanim system in Unity3D. This could be done by introducing a timer that is waiting to receive input from users or simply the movement of knees. The faster the input is received, the lesser the time is accumulated inside the timer, in other words, means the speed is high, thus, the in game avatar should go faster. Related code is a below:

```
public class KinectGes : MonoBehaviour {
        public const int rightKnee = (int)KinectWrapper.SkeletonJoint.RIGHT_KNEE;
        public const int leftKnee = (int)KinectWrapper.SkeletonJoint.LEFT_KNEE;
        //public Text rightKneePos;
        //public Text leftKneePos;
        //public Text speed;

    //public Text diffText;

        private float difference = 0.0f;
        private bool rightLeg = false;
        private bool leftLeg = false;

        private bool startTimer = false;
        private float timerSt = 0.0f;
        private bool stopTimer = true;
        private float timerSp = 0.0f;
    private float animSpeed = 0.0f;

    private float heightBar = 0.09f;
    private Vector3 newPos;
  private KinectManager kinectmanager;

    protected Animator animator;

    // Use this for initialization
    void Start () {
          animator = GetComponent<Animator>();
    }

    // Update is called once per frame
    void Update () {

          if (kinectmanager == null)
          {
              kinectmanager = KinectManager.Instance;

          }

          Vector3 R_KneePos =
kinectmanager.GetJointPosition(kinectmanager.GetPlayer1ID(), rightKnee);
          Vector3 L_KneePos =
kinectmanager.GetJointPosition(kinectmanager.GetPlayer1ID(), leftKnee);

          //rightKneePos.text = "Right Knee Y Position = " + R_KneePos.y.ToString
("F4");
          //leftKneePos.text = "Left Knee Y Position = " + L_KneePos.y.ToString ("F4");

          difference = Mathf.Abs(R_KneePos.y - L_KneePos.y);
          //diffText.text = "Difference = " + difference.ToString("F4");

          if (difference > heightBar && R_KneePos.y > L_KneePos.y && rightLeg == false)
          {
              rightLeg = true;
              startTimer = true;
```

```
            stopTimer = false;
            timerSp = 0.0f;
        }
        else if (difference > heightBar && L_KneePos.y > R_KneePos.y && leftLeg ==
false)
        {
            leftLeg = true;
            startTimer = true;
            stopTimer = false;
            timerSp = 0.0f;
        }

        if (startTimer == true)
        {
            timerSt = timerSt + Time.deltaTime;
        }

        if (stopTimer == true)
        {
            timerSp = timerSp + Time.deltaTime;
        }

        if (difference < heightBar && rightLeg == true)
        {
            animSpeed = 1.0f - timerSt;
            timerSt = 0.0f;
            startTimer = false;
            stopTimer = true;
            animator.SetFloat("Speed", animSpeed); // adjust: right leg animation

            //speed.text = "Speed = " + animSpeed.ToString("F4");
            animSpeed = 0.0f;
            rightLeg = false;
        }
        else if (difference < heightBar && leftLeg == true)
        {
            animSpeed = 1.0f - timerSt;
            timerSt = 0.0f;
            startTimer = false;
            stopTimer = true;
            animator.SetFloat("Speed", animSpeed); // adjust: left leg animation

            //speed.text = "Speed = " + animSpeed.ToString("F4");
            animSpeed = 0.0f;
            leftLeg = false;
        }

        if (timerSp > 1)
        {
            animator.SetFloat("Speed", animSpeed);
            //speed.text = "Speed = " + animSpeed.ToString("F4");
        }

    }
}
```

2. Leaning Left & Right Gestures (Switching lanes)

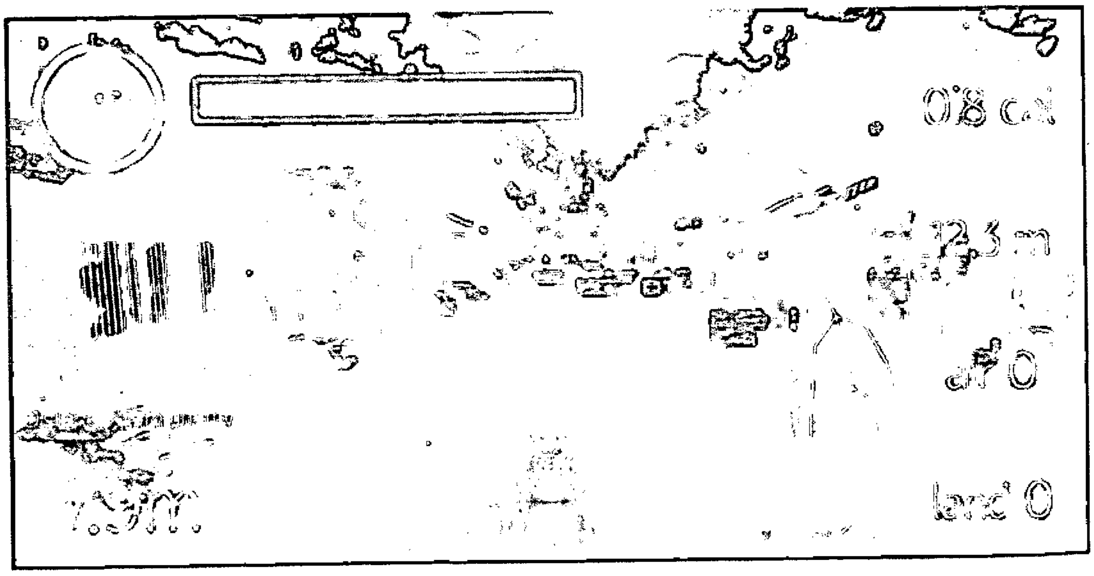


Figure 4.4 In-game avatar changes lane after leaning shoulder gesture is detected

As the game implemented mimicking the features available in those infinite runner themed games, the player will have the capabilities of lane switching in order to avoid obstacles that are coming in their way. By introducing a simple gesture by leaning left or right corresponding to the situation, we provide an intuitive way for user to have some simple workouts. Related code is a below:

```
public class leanLeftRight : MonoBehaviour {

    public const int LeftShoulder = (int)KinectWrapper.SkeletonJoint.LEFT_SHOULDER;
    public const int RightShoulder = (int)KinectWrapper.SkeletonJoint.RIGHT_SHOULDER;
    // Use this for initialization

    private bool switchRight = false;
    private bool switchLeft = false;


    private float tolerance = 0;
    private bool leftIsHigher = false;
    private KinectManager kinectmanager;

    private int Poslocation = 0;
       // Use this for initialization

       void Start () {

       }

       // Update is called once per frame
       void Update () {
        if (kinectmanager == null)
        {
            kinectmanager = KinectManager.Instance;
        }

        Vector3 leftshoulderPos =
kinectmanager.GetJointPosition(kinectmanager.GetPlayer1ID(), LeftShoulder);
```

```
        Vector3 rightshoulderPos =
kinectmanager.GetJointPosition(kinectmanager.GetPlayer1ID(), RightShoulder);



        float difference = Mathf.Abs(rightshoulderPos.y - leftshoulderPos.y);

        if(rightshoulderPos.y > leftshoulderPos.y && difference > 0.05 && switchLeft ==
false && Poslocation >-4){
            transform.Translate(-4, 0, 0);

            switchLeft = true;
            Poslocation -= 4;
        }

        if (difference < 0.02 && switchLeft == true)
        {

            switchLeft = false;
        }

        if (leftshoulderPos.y > rightshoulderPos.y && difference > 0.05 && switchRight ==
false && Poslocation < 4)
        {
            transform.Translate(Vector3.right * 4);
            switchRight = true;
            Poslocation += 4;
        }

        if (difference < 0.02 && switchRight == true)
        {

            switchRight = false;
        }
       }
}
```

3. Raising Arms (Calling Backups)

Figure 4.5 In-game avatar calls backup when raise arm gesture is detected

In confronting the enemies of obstacles inside the gameplay, avoiding or dodging obstacles might exhaust the player especially when enemies comes into the equation, thus, backup for player is implemented to help user in exterminating the enemies as well as destroying the obstacles. In order to call the backup, user would just have to raise his/her arm corresponding to where he/she wants to launch the backup. Related code is a below:

```
public class RaiseHandGesture : MonoBehaviour {
        public bool goLeft = false;
        public bool goRight = false;

        public const int RightHand = (int)KinectWrapper.SkeletonJoint.RIGHT_HAND;
        public const int LeftHand = (int)KinectWrapper.SkeletonJoint.LEFT_HAND;

        public const int LeftShoulder = (int)KinectWrapper.SkeletonJoint.LEFT_SHOULDER;
        public const int RightShoulder = (int)KinectWrapper.SkeletonJoint.RIGHT_SHOULDER;
        private Vector3 initialRightPos;
        private Vector3 initialLeftPos;
        public Text handRaise;

        private bool center = false;
```

3. Raising Arms (Calling Backups)

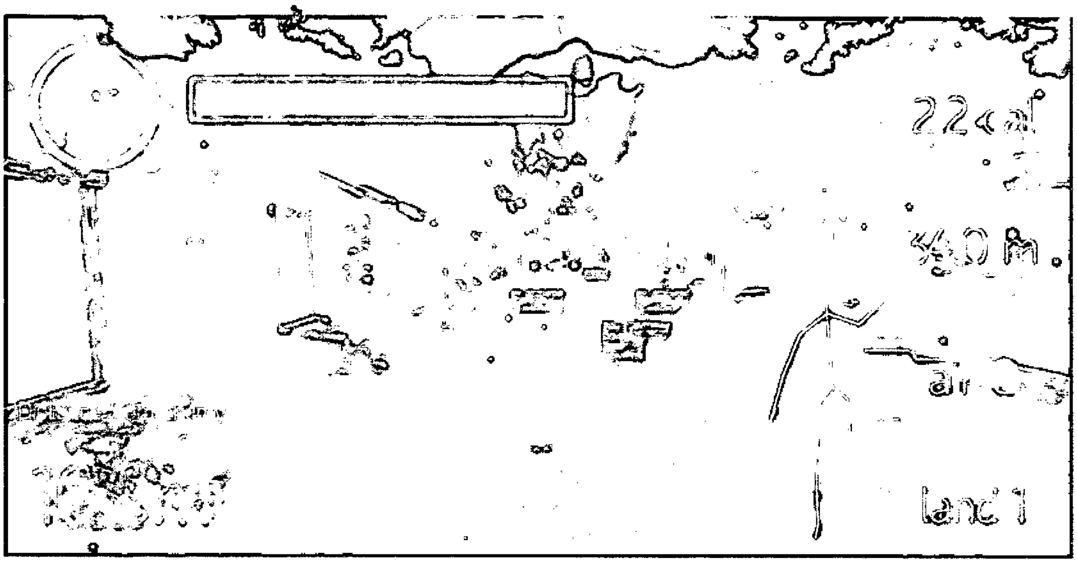


Figure 4.5 In-game avatar calls backup when raise arm gesture is detected

In confronting the enemies of obstacles inside the gameplay, avoiding or dodging obstacles might exhaust the player especially when enemies comes into the equation, thus, backup for player is implemented to help user in exterminating the enemies as well as destroying the obstacles. In order to call the backup, user would just have to raise his/her arm corresponding to where he/she wants to launch the backup. Related code is a below:

```
public class RaiseHandGesture : MonoBehaviour {
	public bool goLeft = false;
	public bool goRight = false;

	public const int RightHand = (int)KinectWrapper.SkeletonJoint.RIGHT_HAND;
	public const int LeftHand = (int)KinectWrapper.SkeletonJoint.LEFT_HAND;

	public const int LeftShoulder = (int)KinectWrapper.SkeletonJoint.LEFT_SHOULDER;
	public const int RightShoulder = (int)KinectWrapper.SkeletonJoint.RIGHT_SHOULDER;
	private Vector3 initialRightPos;
	private Vector3 initialLeftPos;
	public Text handRaise;

	private bool center = false;
	private bool right = false;
	private bool left = false;
	private Vector3 temp;

	// Use this for initialization
	void Start () {
	}

	// Update is called once per frame
	void Update () {
		handRaise.text = "";



		Vector3 leftHandPos = KinectManager.Instance.GetJointPosition
(KinectManager.Instance.GetPlayer1ID (), LeftHand);
		Vector3 rightHandPos = KinectManager.Instance.GetJointPosition
(KinectManager.Instance.GetPlayer1ID (), RightHand);
		Vector3 leftShoulder = KinectManager.Instance.GetJointPosition
(KinectManager.Instance.GetPlayer1ID (), LeftShoulder);
		Vector3 rightShoulder = KinectManager.Instance.GetJointPosition
(KinectManager.Instance.GetPlayer1ID (), RightShoulder);

		CheckRaiseHand (leftHandPos, rightHandPos, leftShoulder, rightShoulder);



	}

	public void CheckRaiseHand(Vector3 lhand, Vector3 rhand, Vector3 lSh, Vector3 rSh){
		if(Mathf.Abs(lhand.y-lSh.y)>0.3f && lhand.y>lSh.y){

			handRaise.text = "Left hand raised";
			goLeft = true;
			MoveLeft();
			goLeft = false;
		}
		else if(Mathf.Abs (rhand.y - rSh.y)>0.3f && rhand.y>rSh.y){
			handRaise.text = "Right hand raised";
			goRight = true;
```

```
                        MoveRight();
                        goRight = false;

                }
        }
}
```

4. Clapping Gesture (Shoot laser)

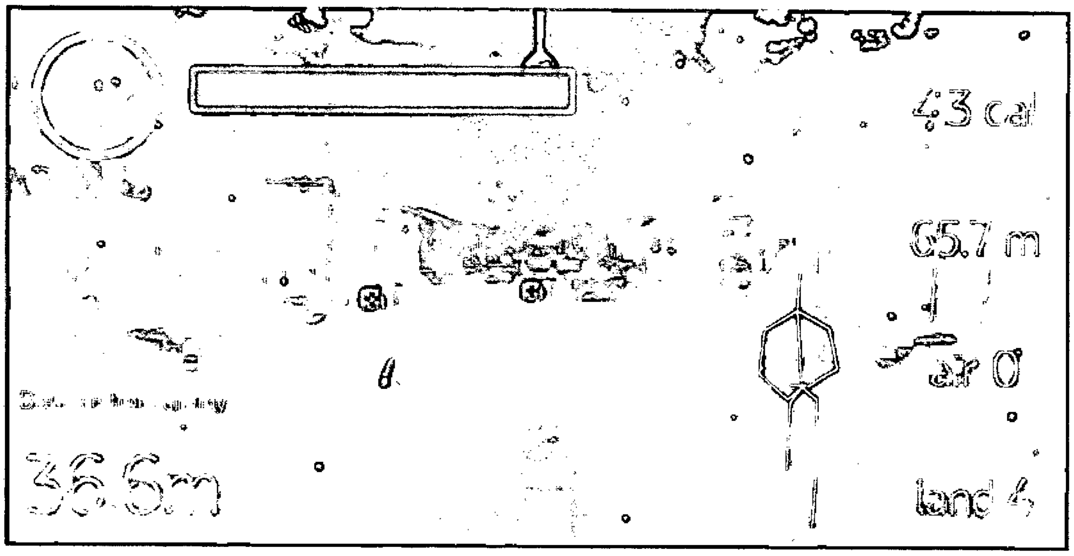


Figure 4.6 Laser is shot when clapping motion is detected while airstrike (air value > 0)

In order to defeat the main enemy who is responsible for deploying its soldier of enemy to apply damage on character, user will need to clap in order to trigger clapping gesture to shoot laser at the main opponent. However, the laser shooting is only applicable if the ammo or airstrike value is not running out. Related code is as below:

```
using UnityEngine;
using System.Collections;

public class DetectClapping : MonoBehaviour {
    public const int LeftHandIndex = (int)KinectWrapper.SkeletonJoint.LEFT_HAND;
    public const int RightHandIndex = (int)KinectWrapper.SkeletonJoint.RIGHT_HAND;

    public const int LeftShoulder =
(int)KinectWrapper.SkeletonJoint.LEFT_SHOULDER;
    public const int RightShoulder =
(int)KinectWrapper.SkeletonJoint.RIGHT_SHOULDER;
    private KinectManager kinectmanager;
    public GameObject lefthandCollider;
    public GameObject righthandCollider;
    LineRenderer line;
    private bool fire = false;
       // Use this for initialization
       void Start () {
        line = gameObject.GetComponent<LineRenderer>();
        line.enabled = false;
       }

       // Update is called once per frame
       void Update () {
        if (kinectmanager == null)
        {
            kinectmanager = KinectManager.Instance;
        }
        Vector3 LhandPos =
kinectmanager.GetJointPosition(kinectmanager.GetPlayer1ID(), LeftHandIndex);
        Vector3 RhandPos =
kinectmanager.GetJointPosition(kinectmanager.GetPlayer1ID(), RightHandIndex);

        Vector3 LShoulderPos =
kinectmanager.GetJointPosition(kinectmanager.GetPlayer1ID(), LeftShoulder);
        Vector3 RShoulderPos =
kinectmanager.GetJointPosition(kinectmanager.GetPlayer1ID(), RightShoulder);

        lefthandCollider.transform.position = new Vector3(LhandPos.x, LhandPos.y,
LhandPos.z);
        righthandCollider.transform.position = new Vector3(RhandPos.x, RhandPos.y,
RhandPos.z);

       }

    public void ShootLaser() {
        StopCoroutine("shooting");
        StartCoroutine("shooting");
    }

    IEnumerator shooting()
    {
        line.enabled = true;
        Ray ray = new Ray(this.transform.position, transform.forward);

        RaycastHit hit;
        line.SetPosition(0, ray.origin);
        if (Physics.Raycast(ray, out hit, 100))
        {
            line.SetPosition(1, hit.point);
            InstantiateExplosion(hit.point);
            InstantiateCollider(hit.point);
        }
        else
        {
            line.SetPosition(1, ray.GetPoint(100));
        }
        yield return new WaitForSeconds(0.5f);
        line.enabled = false;
    }
    void InstantiateExplosion(Vector3 hitpoint)
    {
        GameObject explosion = explosionPooler.explosionpooler.GetPooledObject();

        if (explosion == null) return;

        explosion.transform.position = new Vector3(hitpoint.x, hitpoint.y,
hitpoint.z);

        explosion.SetActive(true);
    }

    void InstantiateCollider(Vector3 hitpoint)
    {
        GameObject explosion = Building1Pooler.b1Pooler.GetPooledObject();

        if (explosion == null) return;

        explosion.transform.position = new Vector3(hitpoint.x, hitpoint.y,
hitpoint.z);

        explosion.SetActive(true);
    }
}
```

```
using UnityEngine;
using System.Collections;

public class DetectHandCollision : MonoBehaviour {
    public Transform sphereleft;
    public bool Fire = false;
       // Use this for initialization
       void Start () {
        Fire = false;
       }

       // Update is called once per frame
       void Update () {
        //Debug.Log("righthand "+transform.position.x + "" + transform.position.y
+ "" + transform.position.z);
        //Debug.Log("lefthand " + sphereleft.transform.position.x + "" +
sphereleft.transform.position.y + "" + sphereleft.transform.position.z);


       }

    void OnTriggerEnter(Collider col) {
        if (col.gameObject.CompareTag("lefthand") && transform.position.x !=0) {
            Debug.Log("clapping");
if(GameObject.Find("PlayerDetails").GetComponent<PlayerDetails>().airstrike>0){
GameObject.Find("Shooter").GetComponent<DetectClapping>().ShootLaser();
GameObject.Find("PlayerDetails").GetComponent<PlayerDetails>().airstrike -= 1;
            }
            Fire = true;

        }
    }

    void OnTriggerExit(Collider col)
    {
        if (col.gameObject.CompareTag("lefthand"))
        {
            Debug.Log("Noclapping");
            Fire = false;

        }
    }

}
```

5. Calculation of Calories Burned

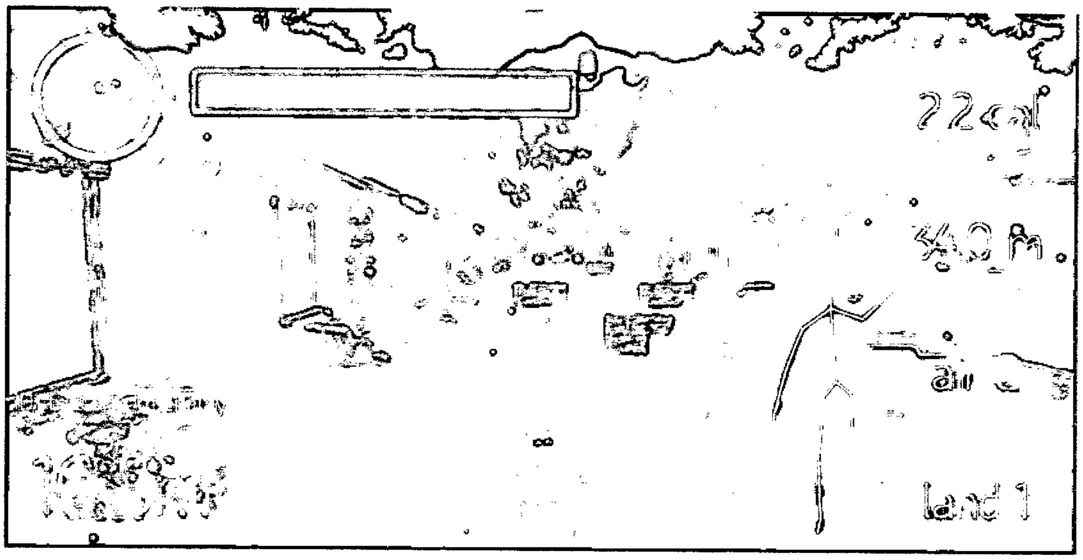


Figure 4.7 Calories and distance reading is available at the right side

5.1 Calculation of Distance in meters

The distance covered by users would be dynamically updated or calculated when the in game avatar moves according to users. In this case, the distance calculation only takes into account of forward movement while ignoring the distance contributed by sideways movement performed by lane switching.

5.2 Calculation of Calories

The calories formula is a simple formula that takes account of a coefficient corresponding to which exercise that user have performed, in this case, it would be jogging/ or running which comes with a coefficient of 0.790; the user' weight as well as the distance he/ she covered in the game. The formula could be written as the following:

Note:

*Weight should be in pounds (1 lbs = 2.2046kg)

**Distance should be in miles (1 miles = 1609.344 meters)

Calories = (jogging or running coefficient) * weight * distance

```
public class CaloriesScript : MonoBehaviour {


        private Vector3 lastPos;
        private double distance;
        private double Calories;
        private double time;
        private double lastDis;
        public Text distanceText;
        public Text CaloriesBurned;
        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
        void Update () {
```

```
        CalculateCalories (CalculateDistance(), 65.0);

    }

    public double Timer(){
        time = (time + Time.deltaTime);
        return time;
    }

    public double CalculateCalories(double distance, double weight){
        //Math.round(calcCoefficient 0.790 * calcWeight * calcDistance)

        double newweight = weight * 2.2046;
        double newdistance;

        newdistance = distance - lastDis;
        Calories = Calories + (0.790 * newweight * newdistance/1609.344);
        lastDis = distance;
        CaloriesBurned.text =  Calories.ToString ("F1") + " cal";
        return Calories;
    }

    public double CalculateDistance (){
        distance = distance + Mathf.Abs(lastPos.z - transform.position.z);
        lastPos = transform.position;
        distanceText.text = distance.ToString("F1") + " m";
        return distance;
    }
}
```

6. Generic GameObject Spawning using Object Pooling Concept

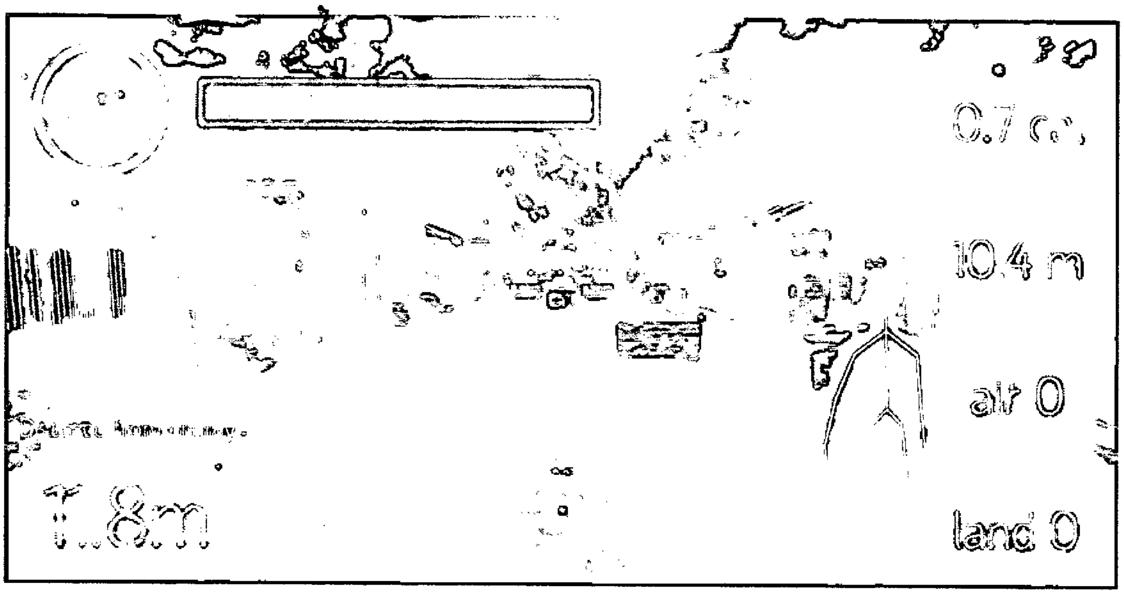


Figure 4.8 The game object such as obstacles, first aid kit and building is generated using object pooling concept

The script or algorithm in this section is responsible for the creation of the game object in real time which acts as the backbone in the level creation of infinite runner game. In other words, the algorithm allows no end of level creation that could be extended corresponding with the player ability or precisely, stamina of that certain player. Related code is a below:

```
public class bulletPooler : MonoBehaviour {
    public static bulletPooler bulletpooler;
    public GameObject pooledObject;
    public int pooledAmount = 20;
    public bool willGrow = true;

    public List<GameObject> pooledObjects;

    void Awake()
    {
        bulletpooler = this;
    }
    void Start()
    {
        pooledObjects = new List<GameObject>();
        for (int i = 0; i < pooledAmount; i++)
        {
            GameObject obj = (GameObject)Instantiate(pooledObject,
transform.position, Quaternion.Euler(0, 180, 0));
            obj.SetActive(false);
            pooledObjects.Add(obj);
        }
    }

    public GameObject GetPooledObject()
    {
        for (int i = 0; i < pooledObjects.Count; i++)
        {
            if (pooledObjects[i] == null)
            {
                GameObject obj = (GameObject)Instantiate(pooledObject,
transform.position, Quaternion.Euler(0, 180, 0));
                obj.SetActive(false);
                pooledObjects[i] = obj;
                return pooledObjects[i];
            }
            if (!pooledObjects[i].activeInHierarchy)
            {
                return pooledObjects[i];
            }
        }

        if (willGrow)
        {
            GameObject obj = (GameObject)Instantiate(pooledObject,
transform.position, Quaternion.Euler(0, 180, 0));
            pooledObjects.Add(obj);
            return obj;
        }
        return null;
    }
}
```

While object pooling algorithm could be generic across the game or scene, the method of using the objects could vary by a lot of degrees. For example, the function or algorithm that used to fire the rocket is not the same as to spawn power ups along the way for user to collect.

6.1 Script used to spawn power ups and obstacles

```
public class spawnWall : MonoBehaviour {
    public int i = 50;
    public int increment = 20;
    public bool finish = false;
       // Use this for initialization
       void Start () {
       }

       // Update is called once per frame
       void Update () {
        Spawn(i);
        if(finish == false){

        i += increment;
        }
        else if(finish == true){
            i = i + 0;
        }

       }

    void Spawn(int posZ)
    {
        GameObject enemy = FirstAidPooler.firstaidpooler.GetPooledObject();
        if (enemy == null) {
            finish = true;
            return;
        }

        finish = false;
        enemy.transform.position = new Vector3(transform.position.x,
transform.position.y, posZ);
        enemy.transform.rotation = transform.rotation;


        enemy.SetActive(true);
    }
}
```

6.2 Script to Spawn Enemies

```
public class FireEnemy : MonoBehaviour {
    public float timelimit = 5f;
    public float timer = 0;
    public float XPos = 0;

       // Use this for initialization
       void Start () {
        //InvokeRepeating("Fire", time, 10);
       }

       // Update is called once per frame
       void Update () {
        timer += Time.deltaTime;
        if(timer >timelimit){
            timer = 0;
            Fire();
        }
       }

    void Fire() {
        GameObject enemy = ObjectPooler.objectpooler.GetPooledObject();

        if (enemy == null) return;

        enemy.transform.position = new
Vector3(XPos,transform.position.y,transform.position.z);
        enemy.transform.rotation = transform.rotation;

        enemy.SetActive(true);
    }
}
```

7. Script for Hover Plane Enemy (Main Enemy)

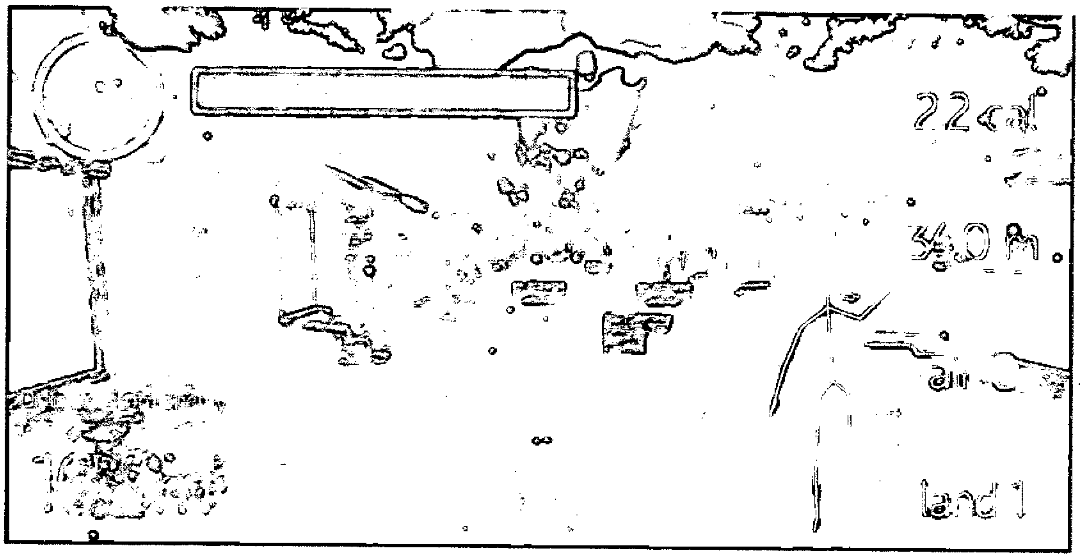


Figure 4.8 Hovering plane with afterburner effect

This is the script used to power the hovering motion of the enemy plane by applying force upwards towards the target with attached rigidbody.

```
Public class HoverPlane : MonoBehaviour {


    public float hoverForce = 100f;
    public float hoverHeight = 1000f;
    private Rigidbody planerigidbody;
```

```
        // Use this for initialization
        void Awake () {
            planerigidbody = GetComponent<Rigidbody>();
        }

        // Update is called once per frame
        void Update () {

        }
    void FixedUpdate() {
        Ray ray = new Ray(transform.position, -transform.up);
        RaycastHit hit;

        if(Physics.Raycast(ray, out hit, hoverHeight)){
            float proportionalHeight = (hoverHeight - hit.distance) / hoverHeight;
            Debug.DrawRay(transform.position, ray.direction, Color.black);
            Vector3 appliedForce = Vector3.up * proportionalHeight * hoverForce;
            planerigidbody.AddForce(appliedForce , ForceMode.Acceleration);
        }

    }
}
```

8. **Input Weight Function**

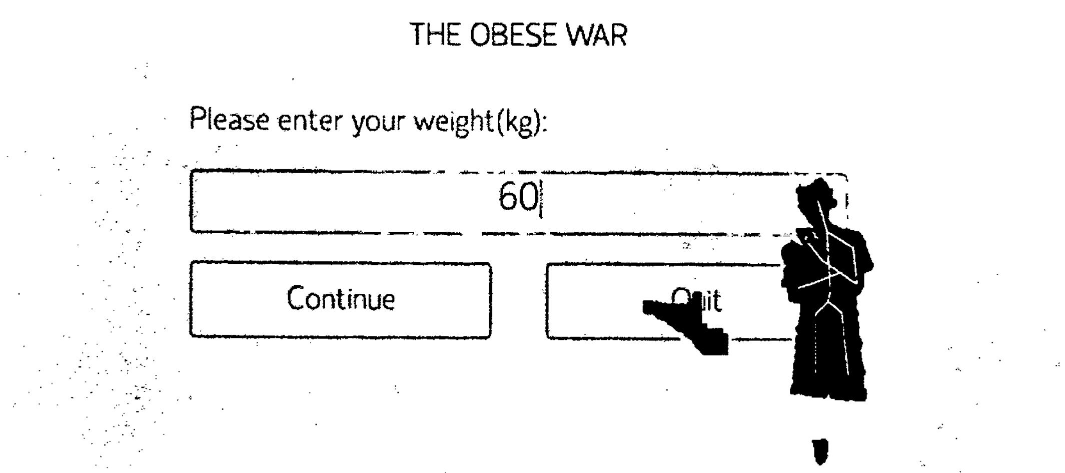


Figure 4.9 Input weight dialogue for calories calculation

Input weight function is required to calculate the calories burned by user in playing this game. Related code is as below:

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;

public class storeWeight : MonoBehaviour {
    public static float weight;
    public float weight2;
    public InputField input;
       // Use this for initialization
    void Awake()
    {
        DontDestroyOnLoad(transform.gameObject);
    }
       void Start () {

       }

       // Update is called once per frame
       void Update () {

       }

    public void returnWeight() {
        weight = float.Parse(input.value);
        weight2 = weight;
        Application.LoadLevel("TestScene");
    }
```

9. GameOver Function

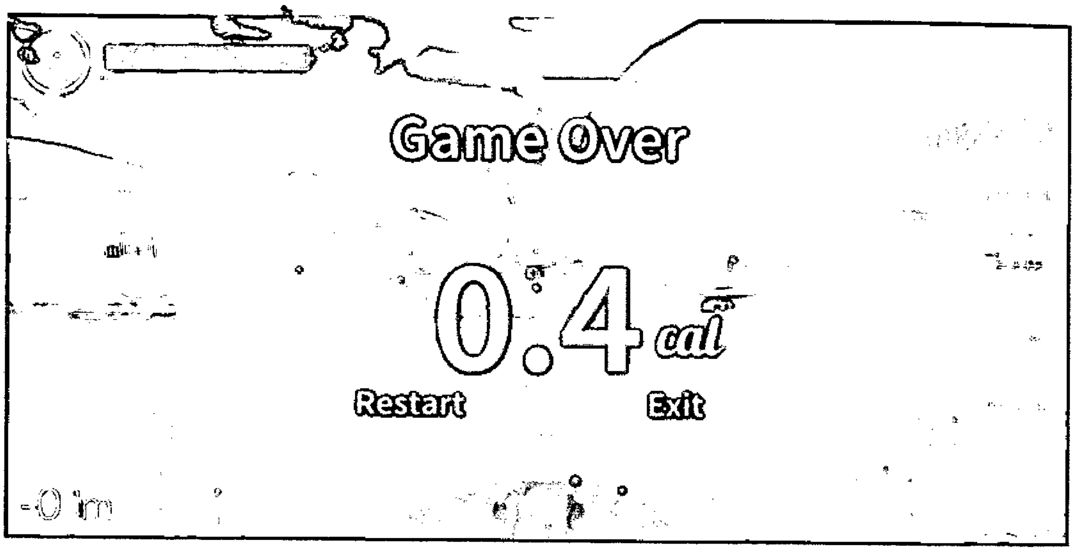


Figure 4.10 Game Over pop out dialogue

Game over function is triggered when the lives of character is equals to zero. In this function, after lives of character reaches zero, the pop out dialog will appear to signal game is over while offering user functionalities to restart or quit the game. Related code is as below:

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
#if UNITY_EDITOR
using UnityEditor;
#endif

public class GameOverScript : MonoBehaviour {

        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
        void Update () {

        }

    public void RestartButton() {
        Application.LoadLevel("TestScene");
    }

    public void Quit() {
        #if UNITY_EDITOR
        EditorApplication.isPlaying = false;
        #else
        Application.Quit();
        #endif
    }
}
```

CHAPTER 5: RESULT AND DISCUSSIONS

5.1 RESULT ANALYSIS

The developed system, Active Game using Microsoft Kinect in Solving Obesity Problem has met all the objectives targeted which are below:

1. To develop a Microsoft Kinect based game to solve obesity problem by promoting active gaming.
2. To create an engaging gameplays to foster interaction between system and users in order to make users committed in performing the given tasks.
3. To develop gameplays that requires both physical as well as cognition interaction from users.

This could be further clarified as below:

1. To develop a Microsoft Kinect based game to solve obesity problem by promoting active gaming.
   - The system or game has been developed based on the capability of Microsoft Kinect Sensor which is skeletal tracking. In this system or game, user is encouraged to exercise, to be active; for example, in order for user to play the game, user would need to walk or run according to their physical abilities and such action would be translated into game as the in game avatar would perform animation of walking or running depends on the speed of user.
   - In addition, other than encourage user to perform cardiac workout in form of brisk walking or jogging as cardiac workout represents the main solution in solving obesity problem apart from proper dieting, the game is integrated with other gestures to cater

the interaction between user and game; for example, gestures such as leaning left or right to switch lanes or calling backups by just raising your left or right arms corresponding to your position and lastly, just clap to shoot a rocket at you main enemy. Such gestures are easy and intuitive for user to perform at ease. In other words, user can start playing this game without any major training.

- Thus, the objective of creating an active game based on Microsoft Kinect Sensor is achieved.

2. To create an engaging gameplays to foster interaction between system and users in order to make users committed in performing the given tasks.
   - If the game is designed just to make user jogs till the end of the level or exhausted, there are very high chances that user will not return to continue his/her regime. It is because simply the gameplay is boring, dull as well as tiring. Thus, one of the objectives of this system is to create an engaging gaming environment that could bring out the fun side of the user to commit in executing the exercise regime. In other words, we want to make exercise feels good, fun and an engaging activity to immerse ourselves in. Speaking of the tasks, all user have to do is to exercise, run through their limit every day, keeps breaking their personal best while playing the game.
   - As to interact with the gameplays, all user have to do is to execute those easy movement or gestures in order for them to control the in game avatar in an infinite runner themed game. In addition, currently recognized gestures are running, lean left or right, raise left or right arm and lastly clapping.
   - In conclusion, an engaging and interactive gameplays is served to user to enhance their engagement as well motivation to continue their exercise regime.
3. To develop gameplays that requires both physical as well as cognition interaction from users.
   - The system started out as an active game that requires physical commitment from the user. However, physical commitment alone will not be enough to sustain user motivation in continuing their exercise regime. Thus, interaction in term cognition should be injected into the game to vitalize the whole system. Rather than just running aimlessly, user would now need to respond to the game environment accordingly or fail the game. With the infinite runner as the main theme in this active game, user will

be challenged by obstacles as well as incoming enemies that is released by the main enemy which could all be destroyed by shooting down the main enemies and there will be collateral damage to the sub-enemies. However, the catch here is that every backups or bullets that is executed is offered at a price, in other words, the distance covered by the user. The further the user run, the more the rewards. Thus, by introducing engaging gameplays alongside an exercise game, cognition interaction is required as to think as well as strategize on how to use the power ups to their advantages. In conclusion, it is like killing two birds with a stone and the objective has been achieved in delivering an active game that emphasize on both physical and cognition ends.

5.2 RESEARCH CONSTRAINTS

Constraints for this project could be categorized into two parts:

1) Development constraints
2) System constraints

5.2.1 DEVELOPMENT CONSTRAINTS

In order to play the active game, user would need to have the Microsoft Kinect Sensor (V1) as there is no support or compatibility offered to Microsoft Kinect V2. Besides, in order to communicate with the hardware of Kinect Sensor, the personal computer of user will need to be equipped with Kinect SDK which could be downloaded freely from Kinect for Windows website. In addition, in order to establish connection of Kinect and PC, user will need to have proprietary USB port as well as a Kinect power adapter to power up Kinect.

In addition, to ensure visual aspect of the game, user will need a dedicated computer monitors or a flat screen TV. Besides, to ensure the active game is lag-free, user will need to have a competent hardware specification with minimum requirement of an Intel i3 powered PC with dedicated graphics card installed to guarantee decent gameplays. Lastly, the user will need a roughly 6 m^2 of space in length of their living room in order for Kinect to accommodate the whole body.

5.2.2 SYSTEM CONSTRAINTS

Currently, only three gestures are coded into the system namely running, lean left or right, raise left or right arms and clapping. Thus, the gameplay would just be limited into using these four gestures to interact with the system. Thus, in other words, the game would be enhanced by integrating more gestures into the system.

1. Future Improvements

The active game could be improved to be a web based Kinect game in order to reach out to broader audience without the need of downloading the application to be installed by the user. In addition, the web based system could be further reinforced by integration of Facebook API to connect potential users around the world. Thus, by enhancing the system to accommodate Facebook Graph and Posts functionality, the active game could be a platform of which obesity patient use to compete in by challenging their friends to new records or even brag their highscores as well as their achievement to others. This could be a step stone to motivate user in performing exercise more often as to compete with other users.

In addition, the gameplay mechanics can be improved by accommodating more features as well as more gestures such as ability to accommodate two person of the same time. Besides, the gestures recognition pool could be expanded to include more intuitively designed gestures that relates to our everyday lives. Thus, making the game more enjoyable and engaging while preserving the user's motivation.

CHAPTER 6: CONCLUSION

The Microsoft Kinect Sensor represents the remarkable advances in computer vision which provides us a limitless potential in term of practical usage at a very affordable price. In this context, Microsoft Kinect is the pivotal element that offers us the marker-less motion tracking technology in developing this system. The algorithm that enables full body skeletal tracking trigger the possibility of the Natural User Interface as coined by Microsoft in which we are offered the liberty to communicate and interact with computer system, console, machine, medical equipment in a whole new way.

The research in this context explores the capability of Microsoft Kinect in recognizing and interpreting gestures to be used as input to interact with the system. Thus, gestures are created and applied into the system to utilize Microsoft Kinect as the input for the active game. Besides, gestures are created or designed specifically to help in cardiac workouts as to ensure user burns more calories while playing the active game. Thus, in this research, efforts were made in designing the gestures corresponding to the capability of skeletal tracking by Kinect in order to promote active gaming among obesity patients.

The result of this research is none other than a system that is cater specifically to obesity patients with engaging gameplay as well as purposeful gestures needed to be performed by the user to help in solving his/her obesity problem. In addition, the system is equipped with calories calculator that responds in real time in order to update user the real time information regarding their progress in term of calories burned; thus, fulfilling the targeted objectives of creating a Microsoft Kinect based game that offers user an interactive and engaging gameplay which requires both user's physical commitment as well as cognition commitment.

In addition, the methodology chosen in this research is none other but waterfall model which enables stable and steady progress from a phase to another phase albeit limiting the flexibility that is offered in other methodology; yet, waterfall guarantees results or deliverables at each phase and most importantly, there is no overlapping of phases which is definitely easier to be managed. Thus, considering the small scale of the research, waterfall is the model to go for.

In conclusion, the research could be enhanced by further exploring the capabilities offered by Kinect such as the features of Kinect Fusion, Facial recognition as well as Voice recognition to be injected into the system in order to create a great gaming environment not only cater the health of obesity patient but health of all people. Lastly, other than exploring Kinect, gadget or device such Leap Motion could be introduce to complement the features that does not come with Microsoft Kinect.

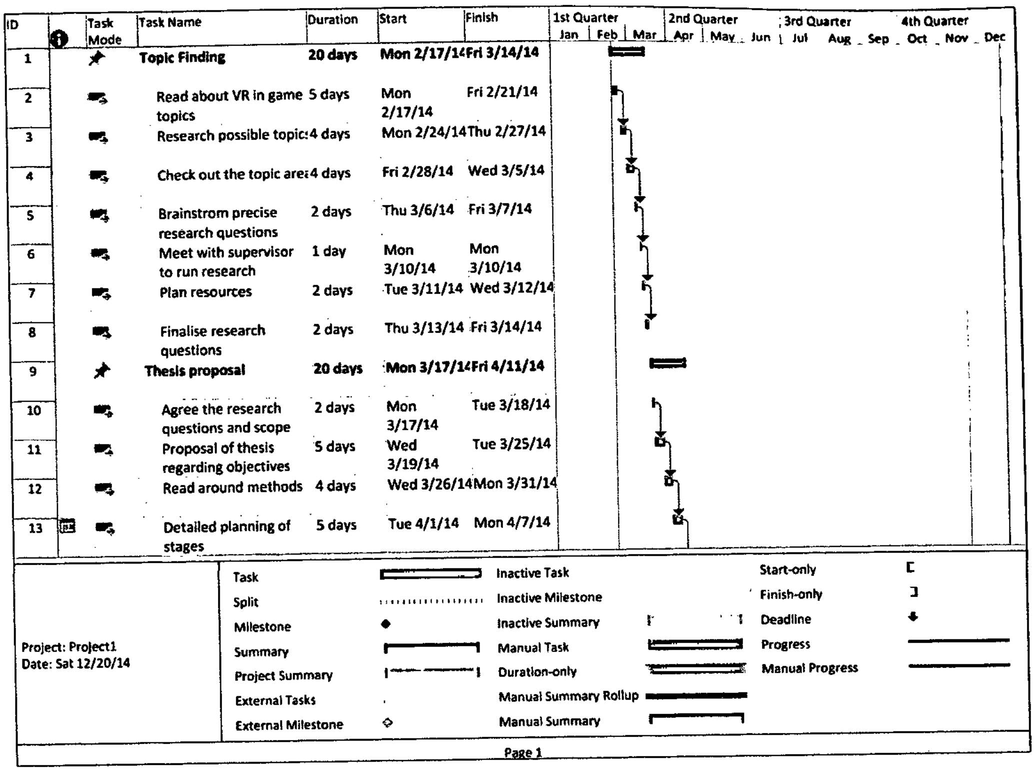
CHAPTER 7: REFERENCES

1. The 10 Fattest Countries in 2014. (n.d.). Retrieved December 7, 2014, from http://www.therichest.com/rich-list/poorest-list/the-10-fattest-countries-in-2014/10/
2. Obesity - INTRODUCTION. (n.d.). Retrieved December 7, 2014, from http://www.clevelandclinicwellness.com/conditions/Obesity/Pages/introduction.aspx#
3. Obesity. (2014, March 30). Wikipedia. Retrieved March 31, 2014, from http://en.wikipedia.org/wiki/Obesity
4. World Obesity Federation | About Obesity. (n.d.). World Obesity Federation | About Obesity. Retrieved April 1, 2014, from http://www.worldobesity.org/aboutobesity/
5. Active gaming. (2014, February 28). Wikipedia. Retrieved April 1, 2014, from http://en.wikipedia.org/wiki/Active_gaming
6. The Future of Children, Princeton - Brookings: Providing research and analysis to promote effective policies and programs for children.. (n.d.). - The Future of Children -. Retrieved March 31, 2014, from http://futureofchildren.org/publications/journals/article/index.xml?journalid=32&articleid=60§ionid=287
7. Are TV and Video Games Making Kids Fat?. (n.d.). Slate Magazine. Retrieved April 1, 2014, from http://www.slate.com/articles/health_and_science
8. Serious game. (2014, March 31). Wikipedia. Retrieved April 2, 2014, from http://en.wikipedia.org/wiki/Serious_game
9. Chien-Yen Chang & Belinda Lange & Mi Zhang & Sebastian Koenig & Phil Requejo & Noom Somboon & Alexander Sawchuk & Albert Rizzo. (2012). Towards Pervasive Physical Rehabilitation Using Microsoft Kinect. Pervasive Health, 159-162. Retrieved from http://ieeexplore.ieee.org.ezproxy.ump.edu.my/xpl/abstractReferences.jsp?tp=&arnumber=6240377&queryText%3Dtowards+pervasive+physical+rehabilitation
10. Anil K. Roy & Yash Soni & Sonali Dubey. (2013). Enhancing Effectiveness of Motor Rehabilitation Using Kinect Motion Sensing Technology. Global Humanitarian Technology Conference: South Asia Satellite (GHTC-SAS), 2013

IEEE, 298-304. Retrieved from http://ieeexplore.ieee.org.ezproxy.ump.edu.my/xpl/articleDetails.jsp?tp=&arnumber=6629934&queryText%3DEnhancing+Effectiveness+of+Motor+Rehabilitation+Using+Kinect+Motion+Sensing+Technology

11. Stephen Simmons & Rachel McCrindle & Malcolm Sperrin & Andy Smith. (2013). Prescription Software for Recovery and Rehabilitation Using Microsoft Kinect. Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference, 323-326. Retrieved from http://ieeexplore.ieee.org.ezproxy.ump.edu.my/xpl/articleDetails.jsp?tp=&arnumber=6563958&queryText%3DPrescription+Software+for+Recovery+and+Rehabilitation+Using+Microsoft+Kinect
12. Exergaming. (2014, April 18). Wikipedia. Retrieved April 21, 2014, from http://en.wikipedia.org/wiki/Exergaming
13. Kinect Sports. (2014, April 17). Wikipedia. Retrieved April 22, 2014, from http://en.wikipedia.org/wiki/Kinect_Sports
14. Kinect Sports Rivals. (n.d.). Kinect Sports Rivals. Retrieved April 22, 2014, from http://www.rare.co.uk/games/kinect-sports-rivals
15. Kinect. (2014, January 12). Retrieved April 22, 2014, from http://en.wikipedia.org/wiki/Kinect
16. Wii Remote. (2014, February 12). Retrieved April 22, 2014, from http://en.wikipedia.org/wiki/Wii_Remote
17. PlayStation Move. (2014, March 12). Retrieved December 5, 2014, from http://en.wikipedia.org/wiki/PlayStation_Move
18. Sony creates a comparison table between Move, Kinect and Wii - Neowin. (n.d.). Retrieved December 8, 2014, from http://www.neowin.net/news/sony-creates-a-comparison-table-between-move-kinect-and-wii
19. Tutorials Point Simply Easy Learning. (n.d.). Retrieved December 8, 2014, from http://www.tutorialspoint.com/sdlc/sdlc_waterfall
20. Calorie calculator. (n.d.). Retrieved December 8, 2014, from http://www.coolrunning.com/engine/4/4_1/94.shtml

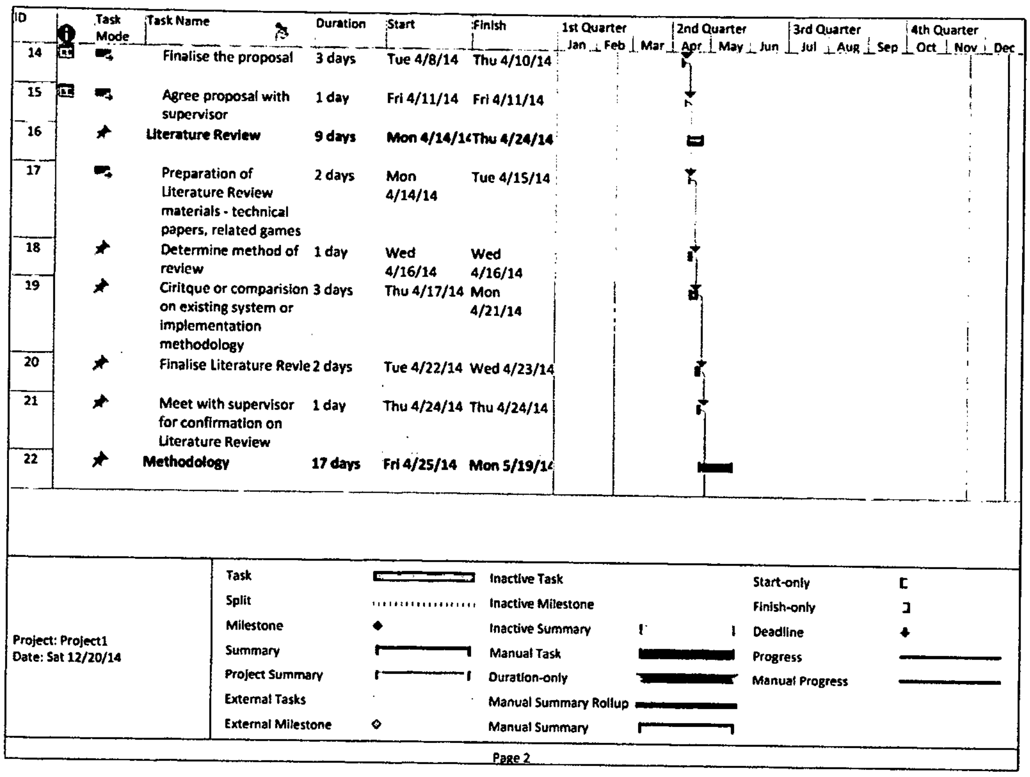
APPENDIX A: GANTT CHART



ID		Task Mode	Task Name	Duration	Start	Finish
1			**Topic Finding**	**20 days**	**Mon 2/17/14**	**Fri 3/14/14**
2			Read about VR in game topics	5 days	Mon 2/17/14	Fri 2/21/14
3			Research possible topic:	4 days	Mon 2/24/14	Thu 2/27/14
4			Check out the topic are:	4 days	Fri 2/28/14	Wed 3/5/14
5			Brainstrom precise research questions	2 days	Thu 3/6/14	Fri 3/7/14
6			Meet with supervisor to run research	1 day	Mon 3/10/14	Mon 3/10/14
7			Plan resources	2 days	Tue 3/11/14	Wed 3/12/14
8			Finalise research questions	2 days	Thu 3/13/14	Fri 3/14/14
9			**Thesis proposal**	**20 days**	**Mon 3/17/14**	**Fri 4/11/14**
10			Agree the research questions and scope	2 days	Mon 3/17/14	Tue 3/18/14
11			Proposal of thesis regarding objectives	5 days	Wed 3/19/14	Tue 3/25/14
12			Read around methods	4 days	Wed 3/26/14	Mon 3/31/14
13			Detailed planning of stages	5 days	Tue 4/1/14	Mon 4/7/14

Project: Project1
Date: Sat 12/20/14

Page 1

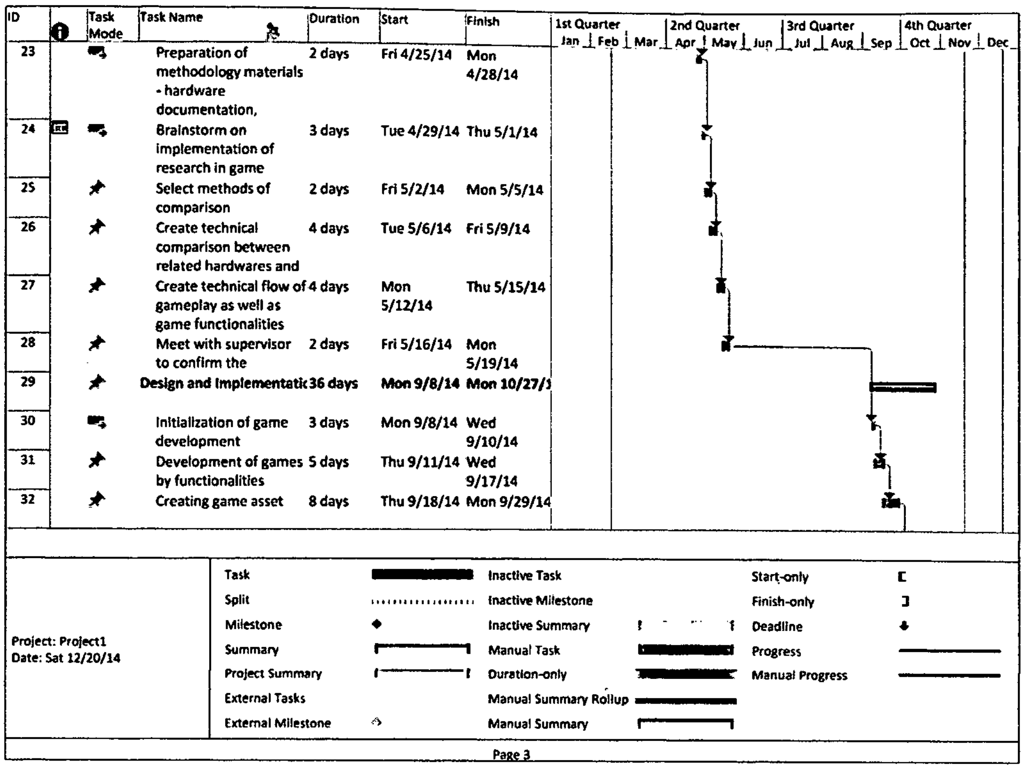


ID	Task Mode	Task Name	Duration	Start	Finish
14		Finalise the proposal	3 days	Tue 4/8/14	Thu 4/10/14
15		Agree proposal with supervisor	1 day	Fri 4/11/14	Fri 4/11/14
16		**Literature Review**	**9 days**	**Mon 4/14/14**	**Thu 4/24/14**
17		Preparation of Literature Review materials - technical papers, related games	2 days	Mon 4/14/14	Tue 4/15/14
18		Determine method of review	1 day	Wed 4/16/14	Wed 4/16/14
19		Ciritque or comparision on existing system or implementation methodology	3 days	Thu 4/17/14	Mon 4/21/14
20		Finalise Literature Revie	2 days	Tue 4/22/14	Wed 4/23/14
21		Meet with supervisor for confirmation on Literature Review	1 day	Thu 4/24/14	Thu 4/24/14
22		**Methodology**	**17 days**	**Fri 4/25/14**	**Mon 5/19/14**

Project: Project1
Date: Sat 12/20/14

Task		Inactive Task		Start-only	[
Split		Inactive Milestone		Finish-only	]
Milestone	◆	Inactive Summary		Deadline	⬇
Summary		Manual Task		Progress	
Project Summary		Duration-only		Manual Progress	
External Tasks		Manual Summary Rollup			
External Milestone	◇	Manual Summary			

Page 2



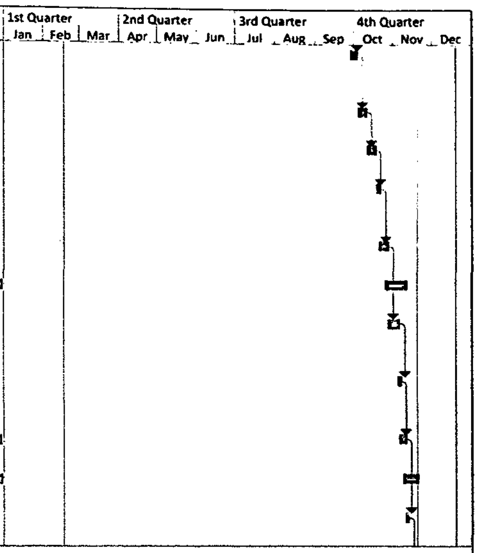
ID		Task Mode	Task Name	Duration	Start	Finish
23		➡	Preparation of methodology materials - hardware documentation,	2 days	Fri 4/25/14	Mon 4/28/14
24	📅	➡	Brainstorm on implementation of research in game	3 days	Tue 4/29/14	Thu 5/1/14
25		📌	Select methods of comparison	2 days	Fri 5/2/14	Mon 5/5/14
26		📌	Create technical comparison between related hardwares and	4 days	Tue 5/6/14	Fri 5/9/14
27		📌	Create technical flow of gameplay as well as game functionalities	4 days	Mon 5/12/14	Thu 5/15/14
28		📌	Meet with supervisor to confirm the	2 days	Fri 5/16/14	Mon 5/19/14
29		📌	**Design and Implementatic**	**36 days**	**Mon 9/8/14**	**Mon 10/27/1**
30		➡	Initialization of game development	3 days	Mon 9/8/14	Wed 9/10/14
31		📌	Development of games by functionalities	5 days	Thu 9/11/14	Wed 9/17/14
32		📌	Creating game asset	8 days	Thu 9/18/14	Mon 9/29/14

Project: Project1
Date: Sat 12/20/14

Task	▬	Inactive Task		Start-only	[
Split	⋯	Inactive Milestone		Finish-only	]
Milestone	◆	Inactive Summary		Deadline	⬇
Summary	┌─┐	Manual Task	▬	Progress	─
Project Summary	┌─┐	Duration-only	▬	Manual Progress	─
External Tasks		Manual Summary Rollup	▬		
External Milestone	◇	Manual Summary	┌─┐		

Page 3

ID		Task Mode	Task Name	Duration	Start	Finish
33			Integrate game asset according to its functionalities	4 days	Tue 9/30/14	Fri 10/3/14
34			Complete game development process	5 days	Mon 10/6/14	Fri 10/10/14
35			Finalize, refine and improve the developed	5 days	Mon 10/13/14	Fri 10/17/14
36			Meet supervisor for confirmation on development	2 days	Mon 10/20/14	Tue 10/21/14
37			Implement the system to get results and data	4 days	Wed 10/22/14	Mon 10/27/14
38			**Results and Discussion**	**11 days**	**Mon 10/27/1**	**Mon 11/10/1**
39			Analyse collected results and intepret it in regards to objective	7 days	Tue 10/28/14	Wed 11/5/14
40			Meet with supervisor to discuss on the intepreted results	1 day	Thu 11/6/14	Thu 11/6/14
41			Compile results into the	2 days	Fri 11/7/14	Mon 11/10/1
42			**Conclusion**	**8 days**	**Mon 11/10/1**	**Wed 11/19/1**
43			Finalize system documentation	2 days	Tue 11/11/14	Wed 11/12/14



Project: Project1
Date: Sat 12/20/14

Task	▭	Inactive Task		Start-only	[
Split	⋯	Inactive Milestone		Finish-only	]
Milestone	◆	Inactive Summary		Deadline	⬇
Summary		Manual Task	▭	Progress	—
Project Summary		Duration-only		Manual Progress	—
External Tasks		Manual Summary Rollup	▬		
External Milestone	◇	Manual Summary			

Page 4

APPENDIX B: TURNITIN REPORT

Turnitin Originality Report

Active Gaming using Microsoft Kinect in Solving Obesity Problem by Vincent Tan Sin Chia

From PSM2 Thesis (PSM2)

Similarity Index	Similarity by Source	
16%	Internet Sources:	8%
	Publications:	4%
	Student Papers:	15%

Processed on 08-Dec-2014 22:11 MYT
ID: 487500002
Word Count: 4863

sources:

1 10% match (student papers from 13-May-2014)
Class: PSM1
Assignment:
Paper ID: 426961156

2 1% match (Internet from 29-Aug-2014)
http://problematic.io/2014/01/

3 1% match (Internet from 22-Feb-2012)
http://blog.almostlogical.com/resources/iTweenFlashReady/iTween.cs

4 1% match (student papers from 30-May-2013)
Submitted to Technological Institute of the Philippines on 2013-05-30

5 < 1% match (student papers from 24-Sep-2014)
Submitted to Columbia Southern University on 2014-09-24

6 < 1% match (student papers from 12-Dec-2012)
Submitted to Universiti Malaysia Pahang on 2012-12-12

7 < 1% match (Internet from 17-Mar-2014)
http://michaelcummings.net/mathoms/creating-2d-animated-sprites-using-unity-4.3

8 < 1% match (Internet from 23-Jun-2013)
http://forum.unity3d.ir/unity3d-t41.html

9 < 1% match (Internet from 01-Feb-2013)
http://blueasa.tistory.com/archive/201211

In this game, running represents the most important exercise that is integrated in this game, in fact, it is the core feature in this game in order to encourage cardiac workout from our target users. In order to track the gesture produced by users, conditions or simply steps have to be introduced into scripting of this gestures. This could be clarified by tracking the movement of running which could be translated into programming as tracking of users' knees positions. In order to enable the recognition of such gesture, computation will be done on users' respective knee position. In this context, the height or y position of knees positions would be compared to ensure walking or jogging steps are executed by users.

After tracking the knees' movements of users, a function or logics would be introduced to compute the speed in order to determine how fast the game avatar should go corresponding to your walking or jogging speed as to blend the animation between jogging and running using the powerful Mecanim system in Unity3D. This could be done by introducing a timer that is waiting to receive input from users or simply the movement of knees. The faster the input is received, the lesser the time is accumulated inside the timer, in other words, means the speed is high, thus, the in game avatar should go faster. Related code is a below:

```
public class KinectGes : MonoBehaviour {
       public const int rightKnee = (int)KinectWrapper.SkeletonJoint.RIGHT_KNEE;
       public const int leftKnee = (int)KinectWrapper.SkeletonJoint.LEFT_KNEE;
       //public Text rightKneePos;
       //public Text leftKneePos;
       //public Text speed;

    //public Text diffText;

       private float difference = 0.0f;
       private bool rightLeg = false;
       private bool leftLeg = false;

       private bool startTimer = false;
       private float timerSt = 0.0f;
       private bool stopTimer = true;
       private float timerSp = 0.0f;
```