3D Sepak Takraw Game

by

Wan Muhammad Nasrullah bin Wan Mansor

Dissertation submitted in partial fulfilment of

the requirements for the

Bachelor of Technology (Hons)

(Information and Communication Technology)

JANUARY 2011

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

3D Sepak Takraw Game

by

Wan Muhammad Nasrullah bin Wan Mansor

11018

A project dissertation submitted to the

Information Technology Programme

University Teknologi PETRONAS

in partial fulfilment of the requirement for the

BACHELOR OF TECHNOLOGY (Hons)

(INFORMATION AND COMMUNICATION TECHNOLOGY)

Approved by,

[illegible]

(Dr. Mohamed Nordin B Zakaria)

UNIVERSITI TEKNOLOGI PETRONAS

TRONOH, PERAK

January 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the reference and acknowledgements, and that the original work contained herein has not been undertaken or done by unspecified sources or persons.

[illegible]

WAN MUHAMMAD NASRULLAH BIN WAN MANSOR

ABSTRACT

Sepak Takraw is a sport which widely popular in South East Asia country, but they are not very popular to the Western countries. Due to this, not much effort has been done in creating a video game version of sepak takraw sport. This fact triggers the idea of this project. "3D Sepak Takraw Game" project is about developing a 3D computer game that simulates the playing of "Sepak Takraw". The main objective of this project is to develop a computer game (prototype) that recreates the playing of sepak takraw sport. The focus of this project is on the graphics and the game physics parts of the game. In this project, a prototype has been developed using C++ language. Graphics part of the prototype was developed with the support of OpenGL API. The prototype successfully implemented several fundamental graphics and game physics techniques. It is also able to demonstrate the basic play of sepak takraw sport. However, there are many other features that the prototype is lacking (e.g. texture, sound, and multiplayer support). There are a lot of improvements needs to be done to make the game more interesting. This dissertation describes the details of this project.

ACKNOWLEDGMENT

The author is indebted and grateful to everyone who has provided both direct and indirect assistance to the completion of this project. Special thanks to Dr. Mohamed Nordin B Zakaria who was the supervisor for this project for all the advices and guidance. Also not to forget my friends who has been supporting me all the time in accomplishing this project. Thanks for the time and effort that has been spent for making this project become a reality. Without them, the project would not become what it is today.

Table of Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Full Meaning
OpenGL	Open Graphics Library
API	Application Programming Interface
OpenAL	Open Audio Library
MSVC++	Microsoft Visual C++
ISTAF	International Sepaktakraw Federation
GLUT	OpenGL Utility Toolkit
IDE	Integrated Development Environment
FPS	Frames per second

CHAPTER 1

1. INTRODUCTION

1.1 Background of Study

This project is about developing a 3D computer game that recreate the playing of sepak takraw sport, thus, it is named as 3D Sepak Takraw Game. The area of interest in this game development is about the implementation graphics and animations methods in simulating the sport. This is quite challenging because there is no sepak takraw computer game that currently available on the market. The implementation of this project will be represented by the game prototype which expected to be built on C++ language with the support of OpenGL API. The prototype will only demonstrate the very basic features of a typical sport game.

1.2 Problem Statement

Currently, many sports have been recreated as computer games. For example, FIFA series by EA Sports and Pro Evolution Soccer series by Konami Digital Entertainment for soccer, NBA Live by EA Sports for basketball, and NHL series by EA Sports for ice hockey. However, it is almost impossible to find sport game that simulates the sepak takraw sport. Based from my research, the only Sepak Takraw simulation game that I was able to find is "Deca Sports DS" by Hudson Soft. Yet, it is actually a collection of several mini games. It consists of 10 games of different sports (Goergen, 2010) – sepak takraw is one of them. Though, the game was not well designed and it is only for Nintendo DS console – not for PC.

The game developers seem not really bothered to make a game that recreates the playing of sepak takraw. The most possible reason is because of the sport game developers always focused on developing games for widely known sports because of its marketability – especially those prominent sport game developers like Electronic Arts Sports, Konami Digital Entertainment and 2K Games.

Due to above issues, there is not much works are done in recreating the sepak takraw sport as computer game. Following this matter, it triggers the idea of this project – to develop a 3D Sepak Takraw game.

1.3 Objectives

Followings are the objectives of this project:

1.3.1 To study and apply fundamental computer 3D graphics in modeling the sepak takraw game

1.3.2 To study and apply basic game physics in simulating the sepak takraw game

1.3.3 To develop a computer game (prototype) that recreates the playing of sepak takraw sport

1.4 Scope of Study

This project is mainly focused on the graphics and animations parts of the game since they are likely the most complex part of the game. All graphics and animations that appear in the game – like the movement of the player and the ball – must be carefully studied to attain the best visual simulation.

Besides that, the next thing that needs to be considered is the game physics. It is important to study the physics of the game to make the game appears to be even more realistic to the user. For instance, the frictions, the gravity and the ball trajectories, all of these factors must be precisely calculated by the game physics engine to get the most sensible gameplay.

CHAPTER 2

2. LITERATURE REVIEW

This chapter will describe several theories related to the game design and information about sepak takraw.

2.1 Sepak Takraw Overview

Sepak Takraw is a sport that originated in Southeast Asia. The name of the game comes from the words Sepak, a term used in Malaysia, Singapore and Indonesia which translates into "kick", and Takraw, a term used in Thailand which means "woven ball" (Hurst, 2005). In English, sepak takraw is known as "kick volleyball".

In an article titled, "DECA SPORTS DS - History of Sepak Takraw", Justin Vonderach describe sepak takraw as a hybrid sport mixing both Soccer and Volleyball. It is similar to volleyball in sense that the ball is volleyed over a net until it touches the ground. It is also similar to soccer for the fact that only feet, knees, chest and head are allowed to touch the ball.

The origins of sepak takraw can be traced back to 15th century Sultanate of Malacca. During that time Sepak takraw is known as "sepak raga". It is considered as traditional game played by local people. Originally, the sport was played in open area only by the royal family. Later, it was spread to all local people.

Sepak raga was played in a team whereby they play the ball in a circle. The game requires skills in volleying and controlling the ball. In earlier times, the ball was made of rattan which was can easily obtained locally. Though, with the technology advancement, the ball is now made of plastic.

In 1962, sepak raga started to be played on the court with net. The number of player has fixed to three for each team. Nowadays, sepak takraw is played by all people in South East Asia. It is officially recognized by several countries and has become a part of SEA Games in the year of 1965. The sport name also has been changed to "Sepak Takraw".

The play of Sepak Takraw international is controlled by the ISTAF (International Sepak Takraw Federation). Basically, the current way of playing Sepak Takraw consists of two sides with three players in each team. The first side serves the first ball to start the first set. The victor of the first set then has the choice of choosing who servers next. After the serve, players are free to move about their court within the boundaries. If a team commits a fault, the point along with service is awarded to the opposing team. Whichever team reaches 21 points first wins. Any use of the hands will result in change of service.

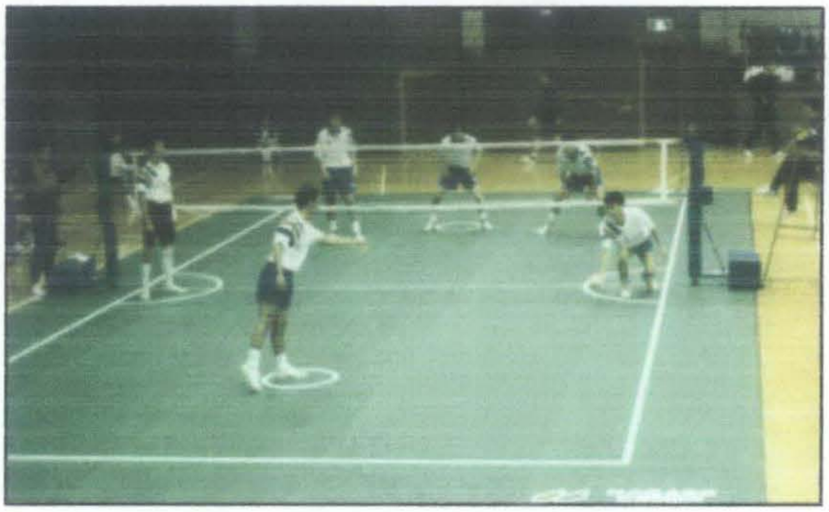


Figure 1: Sepak takraw match

2.2 Graphics

As mentioned earlier, the focus of this project is on the implementation of the graphics and animations in simulating the sport. Graphics plays the most significant part on most of modern games. It directly reflects the overall rating a game. The game graphics referred here also include the game animations. There are many methods of creating a game graphics. Each method has its own advantages and disadvantages.

For this project, only basic graphics techniques will be implemented to narrow down the project scope. However, this project still working 3D graphics instead of 2D graphics.

2.2.1 Graphics API (DirectX vs. OpenGL)

"DirectX vs. OpenGL" has been a common topic of discussions among developers when it comes to graphics programming arena. Both API has its own advantages and disadvantages. Undoubtedly, most of computer games available on the market use Microsoft DirectX API in developing the game (including the graphics part). This is because of the fact that DirectX is easier to be used for games graphics development. It is also packaged with other APIs (e.g.: DirectSound, DirectInput).

However, for this project, OpenGL API will be used in developing the graphics. Unlike Microsoft DirectX, OpenGL is open source. Moreover, there is a toolkit known as GLUT (OpenGL Utility Toolkit) which is widely distributed on the internet that makes the graphics development work using OpenGL simpler. In term of complexity, OpenGL draw call cost is lower than Direct3D, hence, resulting in better performance (Green, 2006). In addition, there are many OpenGL tutorials available on the internet making it a better choice for graphics programmer with low experience.

2.2.2 Player Representation

In this project each player will be represented as an articulated figure. Articulated figure only represent the major parts of the model. It is easy to be animated and commonly used by video game artists and in the movie industry.

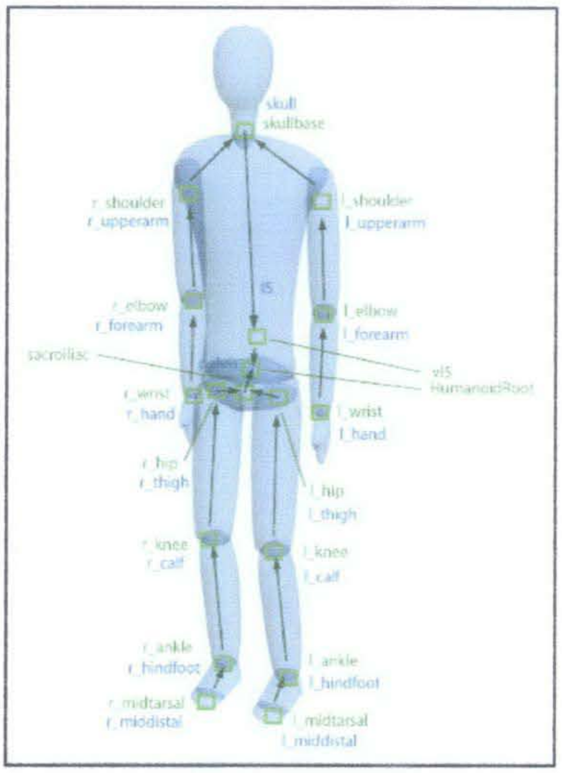


Figure 2: An example of articulated figure

Kiss (2002) explained the articulated figure as a figure which consists of rigid links known as segments in H-Anim terminology – is a standard for articulated human figures (Web3D Consortium, 1999). Those segments are connected by joints with 1, 2 or 3 Degrees Of Freedom (DOFs). The control methods may use kinematics (time-based joint rotation values) or dynamics (force-based simulation of movement) to drive the articulated figures, methods for which the concepts and basics are presented in many animation textbook.

Simulating the player animations is the hardest part in graphics programming for this game. To get the most realistic animation, the movements of the real sepak takraw player will be used. The movement of the real player will be captured (motion capturing) and stored as key frame data – i.e. array. Interpolation of data may be needed between the key frames to get a more fluid animation.

Traditionally, computer animation data is stored as key-frame data. This data structure has two components: the key which specifies relative time moments (or frame) and the animation data values at that particular time (e.g. the angle of rotation data). Getting these data values can be done manually (by talented animators) or can be captured automatically using different types of motion capture hardware like body motion sensors. However, the motion capture device is very rare and expensive.

2.2.3 Environment Representation

The environment here refers to the background and all objects around the court area – including the court itself. The environment must also be properly designed so that the game prototype can deliver the most realistic gameplay to the user. All environment objects that are fixed to the game law will be designed according to the game law.

Followings are the extracted important environment objects information that can be useful to the project:

- **The Court**

  Figure 3 below show the official dimension of sepak takraw court.

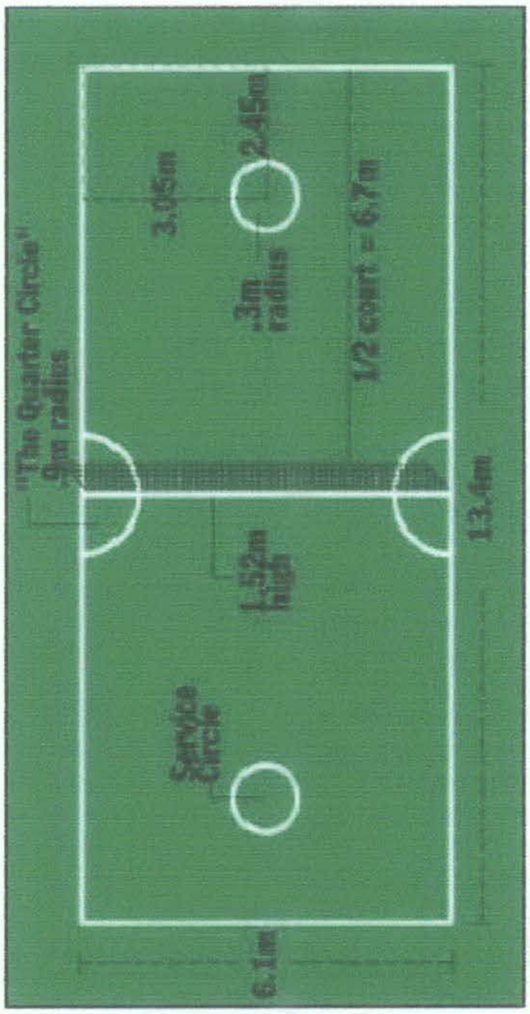


Figure 3: Official Sepak Takraw court layout

- **The Net Posts**

  Post height: 1.55m

  Post Radius: 0.04m in radius.

  Post Position: 0.3m away from the sideline and in line with the Center Line

- **The Ball**

  Ball circumference: 0.42m - 0.44m

2.3 Sound

Compared to graphics, the sound is less significant to the user. It is also not really interested in this project domain. Nonetheless, it is still good to considered about it and implemented it. Normally, in a sport video game, there are at least two types of sound. They are the game sound and the background sound. Taking a soccer game for example, the game sound refers to all on the pitch sounds which includes the ball sound and the referee whistle, while the background sound may refers to the sound made by the crowds or the game background music. For this project, only the game sound will be considered – which is the ball sound effect.

The sound (audio) part of this project will be developed with the support of OpenAL (Open Audio Library) API.

2.4 Game Physics

The game physics is another important point to be considered in this project. For the best result, the game prototype must implement all necessary physics elements so that the simulation will appear to be logic (realistic). For this project, the object that heavily applies the physics is the ball. The ball requires a lot of physics factors to be considered, for example, the ball speed, acceleration, friction, and elasticity.

2.4.1 Bouncing Ball Physics (Gravity)

There is an article written by Helmut Knaust explaining about the basic of bouncing ball physics. Knaust said that when a ball is subject to free fall, at time (t) the ball will be at height (h) if g is the gravity, given the following formula:

$$\boxed{h(t) = h - \frac{g}{2}t^2}$$

Figure 4: Basic formula ball bouncing physics

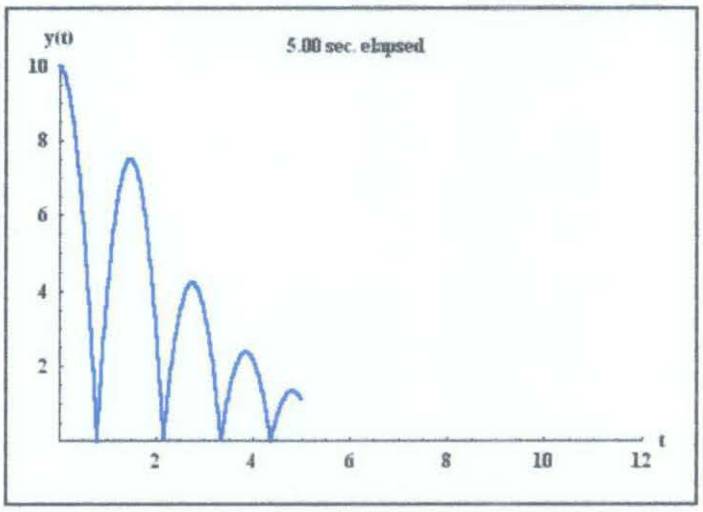


Figure 5: Example of a graph that shows the height of the ball, y(t), against time, t

2.5 Gameplay

Since there is no sepak takraw game for PC available on the market, the gameplay for this project prototype will emulate the gameplay of volleyball game (which widely available on the market). The 'gameplay' here mostly refers to the player's controls. Volleyball game is chosen because is about the same as sepak takraw game. The only significant different is sepak takraw does not allow the players to use their hands.

At the moment, I will be using "Klonoa Beach Volleyball" game as the guideline in developing the gameplay for this project. Klonoa Beach Volleyball is a 3D side-view beach volleyball game by "Namco Limited". This game is chosen because of its simple gameplay which make it easy to be emulated on for this sepak takraw game prototype. Despite its gameplay simplicity, the arcade-like gameplay of Klonoa Beach Volleyball is still enjoyable to be played.

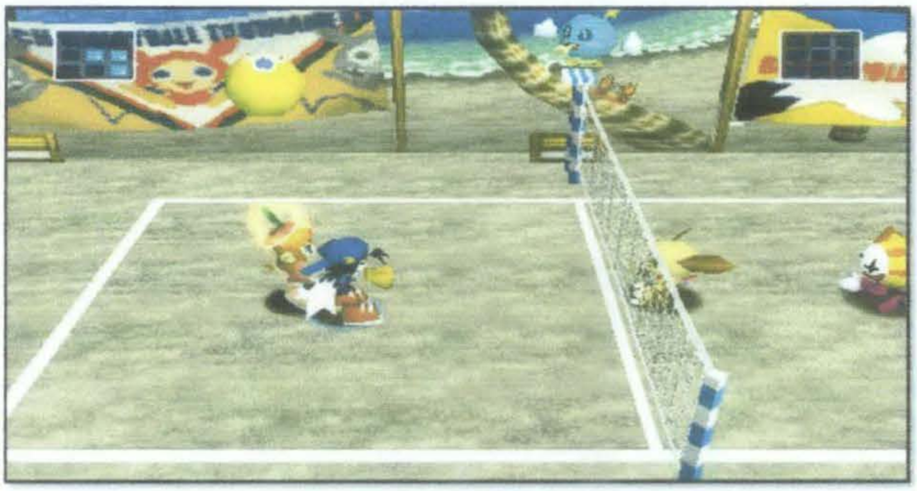


Figure 6: Klonoa Beach Volleyball

2.6 Game Rules and Regulation

The official sepak takraw sport rules and regulation will be used for this sepak takraw game. The official rules and regulations are maintained by International Sepak Takraw Federation (ISTAF). It can be freely obtained from the internet (http://www.sepaktakraw.org/istaf.html). Any matters concerning about the game law will be referred to this document.

CHAPTER 3

3. METHODOLOGY

A prototype will be developed for this sepak takraw game. The prototype will implement all suitable theories and methods discussed previously in Literature Review chapter. In other words, the prototype will represent the result of this project. The prototype will be focused on demonstrating the use of graphics and game physics that in simulating the sepak takraw game. This chapter describes the details of the methodology and the project works of the game prototype development.

3.1 Development Model

The prototype development will progress based on Iterative and Incremental Development Methodology. The advantage of using iterative & incremental methodology is that each cycle ends with a usable system/prototype. The development can be stopped without the entire project being abandoned (Potts, 1998). This makes this methodology is suitable for a research project like this project – given the fact that this project might be continued by other people later. Aside from allowing the developer to try to implement new features at each cycle, this methodology also makes rolling-back work easy in the event of error.

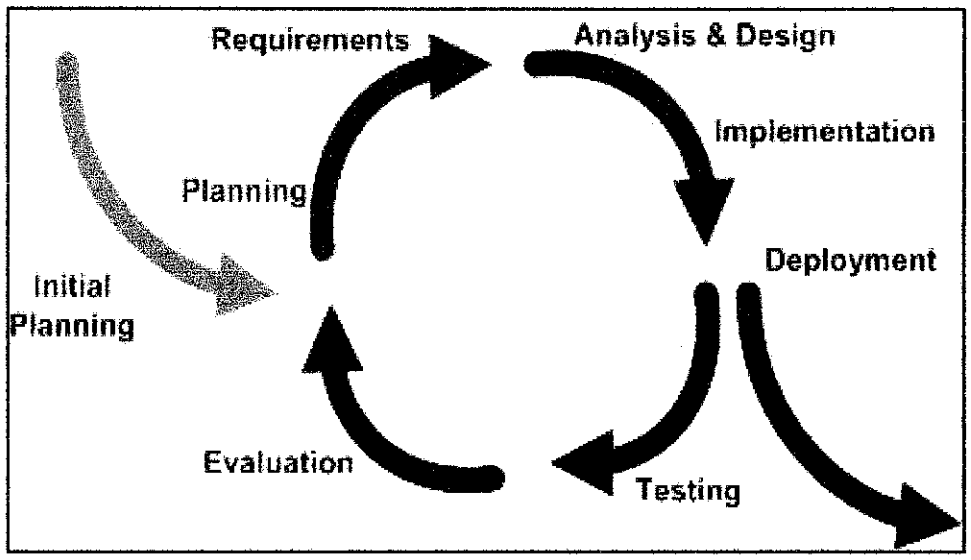


Figure 7: A typical Iterative and Incremental Development Model cycle

The basic principle of this methodology is to develop a prototype through repeated cycles (iterative) and in smaller portions at a time (incremental). Each cycle involves analysis, redesign, implementation and testing phase whereby new features are added and some changes in the design are made. For this project, each successful cycle represents one key milestone accomplishment. The key milestones for this project will be discussed later in this chapter.

3.2 Tools

The game prototype will be developed on C++ language with the support of OpenGL API. C++ is a statically typed, free-form, compiled, general-purpose programming language. It is regarded as an intermediate-level language, as it comprises a combination of both high-level and low-level language features (Schildt, 1998). It was chosen for this project because it allows the use of class that suits object-oriented programming. It also has a good support for OpenGL. There are many OpenGL examples written in C++ on the internet.

OpenGL (Open Graphics Library) is a standard specification defining a cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. OpenGL was developed by Silicon Graphics Inc. (SGI) in 1992 and is widely used in computer aided drawing (CAD), virtual reality, scientific visualization, information visualization, and flight simulation. In this project, OpenGL will be used in developing the graphics part of the game prototype.

These are the tools that required for the prototype development:

- **Microsoft Visual C++**

  Microsoft Visual C++ is an Integrated Development Environment (IDE) by Microsoft for C/C++ languages. It is integrated with tools for developing and debugging C/C++ code. Therefore, it is very good C/C++ programming tool especially in developing code written for the Microsoft Windows API.

- **GLUT (OpenGL Utility Toolkit) API**

  GLUT is a window system independent toolkit for writing OpenGL programs that implements a simple windowing API for OpenGL. GLUT is designed for constructing small to medium sized OpenGL programs, hence, making it suitable for this project. GLUT also provides a portable API that allows OpenGL program works across all PC and workstation OS platforms.

- **OpenAL API**

  OpenAL (Open Audio Library) is a free software cross-platform audio API. It is designed for efficient rendering of multichannel three dimensional positional audio. Its API style and conventions deliberately resemble those of OpenGL. OpenAL API is suitable for use with gaming applications and many other types of audio applications.

3.3 Key Milestone

Followings are the main activities of the project in sequence:

- Create player model
- Animate player model
- Create the environment model
- Implement the ball physics
- Implement the game flow and controls
- Implement AI (Artificial Intelligence)
- Implement the sound

3.4 Project Activities

Basically, this project is about applying the techniques, theories and information discussed earlier in the literature review to develop the game prototype. To be more specific, the projects activities are indentified to break down the project. The time and effort for each activity has been estimated and allocated carefully based from my previous experience. This is important to ensure the project able to be finished on time. Please refer to the Gantt chart below for more detail about the project activities.

	January 2011 Semester													
Activities / Week #	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Create player model	X	X												
Animate player model		X	X											
Create the environment model				X										
Implement the ball physics				X	X									
Implement the game flow and controls					X	X								
Implement AI (Artificial Intelligence)						X	X							
Implement the sound								X						
Refinement / Bug Fixing									X	X	X			

Figure 8: Prototype Development Gantt chart

CHAPTER 4

4. RESULT & DISCUSSION

This chapter will discuss about the result of this project which is the prototype of the game. Please refer to Appendix I for the screenshot of the game prototype. Below is the summary of the project status.

Activities	Status
Create player model	Done
Animate player model	Done
Create the environment model	Done
Implement the ball physics	Done
Implement the game flow and controls	Done
Implement AI (Artificial Intelligence)	Done
Implement the sound	Dropped

Table 1: List of project activities

Apparently, during the project development, the sound part was taking more time than expected. Therefore, I decided to drop the game sound feature the project due to this time constrain issue. Nevertheless, the other parts of the project were successfully completed.

4.1 Graphics

The graphics part of the game prototype has been fully completed. The game window aspect ratio is set to 16:9 – a standard wide screen ratio – and running at the resolution of 1152 x 648. All modern computers should not have any trouble to run at this resolution. All the game objects are also properly scaled down to the ratio of 1 cm per pixel to ensure uniform object scales.

The graphics part of the game prototype has been fully completed. The game window aspect ratio is set to 16:9 – a standard wide screen ratio – and running at the resolution of 1152 x 648. All modern computers should not have any trouble to run at this resolution. All the game objects are also properly scaled down to the ratio of 1 centimeter to 1 pixel to ensure uniform object scales.

4.1.1 Player Model

The player model is the most complex model of the game. The player model of the game prototype is actually a model human articulated figure. In this prototype, the Human Articulated Figure consists of two types of parts which are the Joints and the Segments. In this prototype, a Segment is drawn as a box while the Joint is drawn as a sphere. A Segment can be attached to a Joint, and Joints can be attached to a segment. The result is a chain of several Segments and Joints – This is done in C++ by making use of linked list technique.

The most significant different between the Segment and the Joint is that the Joint is rotatable. The joint rotation data can be stored and applied to the joint whenever necessary. If a set of joint rotations data is applied to a model human articulated figure – which have several joints – it will make the human figure to perform a certain pose. Therefore, a set of joint rotations data is stored in a vector as Pose data.

Applying Pose-to-Pose Animation technique, if several Poses data is applied to an articulated human figure rapidly, it will make the human figure appears as if it is performing an action. Thus, a collection of Poses data – stored in a vector – is called as an Action. Therefore, an action is actually a huge collection of Joint rotation data. Please refer to Appendix II for example of the player Poses data.

Motion capture refers to the work of deriving the Joint rotation data from a real world player performance. Motion capture is not an easy task since it involves a huge amount of Joint rotation data. To simplify the motion capture work, only key frame data (i.e. key Pose data) are captured. Then, the intermediate frames between the two key frames are generated by interpolating the key frames data. In this prototype, only the key Pose data are derived and stored as the game data. The intermediate Poses data are generated once the game is started.

For this prototype, there are four types of motions that have been captured from the real sepak takraw player:

- Move up/down
- Move left/right
- Instep kick
- Bicycle kick

4.1.2 Environment Model

There are three environment objects for this game. They are:

- The Ball
- The Court
- The Net

The ball is represented by a sphere. The real/official dimension of a sepak takraw ball is used to draw the sphere. The rule of speak takraw said that the ball circumference must be 44cm, which means 7cm in radius. Following the pixel per centimeter, the 7cm means 7 unit of pixel. Thus the sphere representing the ball radius is set to 7. The ball also has spinning effect to create an illusion that make ball flight appears to be more realistic. This is done by applying rotation transformation to the ball according to ball vector.

The court and net dimensions also follow the official rule of sepak takraw. According to the official rule of sepak takraw, the court dimension must be 610cm wide and 1340cm long while the net post must be 155cm height. Scaling these values into the game prototype coordinate system makes the court to have a dimension of 610 by 1340 while the net to have a height of 155. Both the court and net are represented as boxes. The net is made to be appeared transparent to allow the user to see the opposition part of the court.

4.1.3 Viewing mode

To make the prototype development work easy, there are two mode of viewing implemented in this prototype. They are orthographic mode and perspective viewing mode. The perspective view will be the default mode for playing the game. Perspective view allows a better perception of three dimensions objects thus offering a better gaming experience. The orthographic mode is just an added feature and was used mainly to assist me during the game development.

4.1.4 Animation Speed Control

There was an issue regarding the animation speed. The speed of animation is not consistent. The animation is heavily depending on the situation – depending on the complexity of the calculation performed at that moment and the machine performance. The frame rate fluctuates from 140 FPS (frames per second) to over 300 FPS. To overcome this problem the frame rate must be controlled (throttled) to ensure a consistent frame rate. Usually, anything higher than 60 FPS is considered good. Thus, for this game, the frame rate is locked at 120 FPS to ensure consistent frame rate. This eliminates all jitter or game speed related issues encountered before.

4.1.5 OpenGL 2D Text

There are some parts of the game that use 2D text, for example, the game score. There are many ways to implement 2D text in OpenGL. The easiest way is by using GLUT font rendering functions. GLUT supports two types of font rendering:-

- Stroke fonts
- Bitmap fonts

In Stroke fonts, each character of the text is rendered as a set of line. Segments, while in Bitmap fonts each character is a bitmap generated. Both methods can be easily used – by calling glutStrokeCharacter function or glutBitmapCharacter function.

Bitmap fonts method will result in better looking text but it require a lot of processing power, thus, affecting the overall game performance. On the other hand, the Stroke fonts method is faster but the text does not look as fine as Bitmap fonts. I ran a test to measure the performance of both methods. From the test results, it appears that using Bitmap fonts caused the game frame rate dropped from 120 FPS to around 70 FPS, while using Stroke fonts almost does not affects the game performance at all. For these reasons, I decided to use Stroke fonts method from the game prototype.

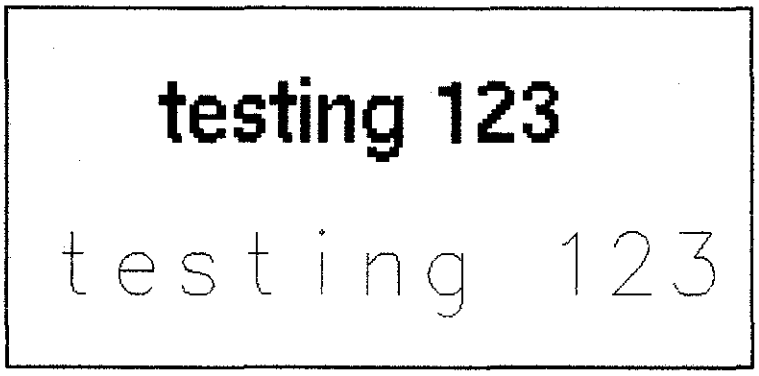


Figure 9: Comparison using bitmap font (above) against stroke font (below)

4.2 Game Physics

The game physics is another element focus of this project. The game physics generally require a lot of mathematical calculations to achieve accurate physics effects. During the development, several game physics were studied and implemented into the game prototype. In this game, the game physics were mainly applied to the ball. As the result of the game physics implementation, the game prototype looks more realistic. However, since the speed of simulation (game speed) is more important than accuracy of simulation, this leads me to designs for physics engines that favor speed over the physics accuracy – by using several approximation techniques.

4.2.1 Collision Detection

Collision detection typically refers to the computational problem of detecting the intersection of two or more objects. In this game, it is used to detect the situation when the ball touches other objects – i.e. the player or the court – in the game. Solving collision detection problems requires extensive use of concepts from linear algebra and computational geometry.

There are many methods available that can be used to detect the collision. In this project, Bounding Volume methods were used. In this method, the solid objects in the game are bounded with imaginary boxes or spheres. By using several mathematical calculations, the intersection between two or more bounding can be detected.

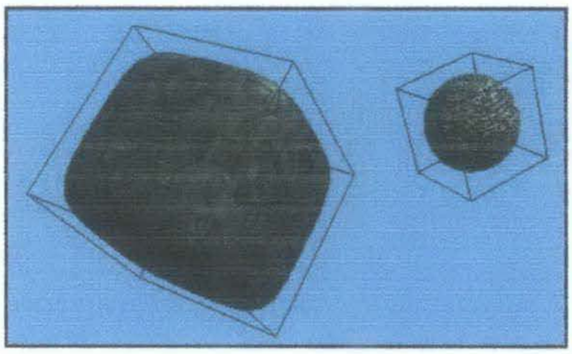


Figure 10: Bounding box example

In the game prototype, every single objects (ball, players, net and court) of the game are bounded with a bounding. Then, all those boundings are added into a list. In each frame, all the boundings in the list are checked for intersection. If there is an intersection, it means that there is a collision between those objects that are bounded to those boundings.

4.2.2 Ball Physics

Most of the game physics are implemented on ball object. The first one is about vector. Vector is a geometric entity that comes with both length and direction. In physics, vectors are typically used to represent physical quantities which have both magnitude and direction. In this project, the vector is one of the fields for the object ball. The vector represents the direction and speed of the ball. The vector in this case consists of three more fields (stored as floats) that represent the values of the x, y and z components of the vector. The ball will move according to the vector. In each frame, the current ball position is added with the ball vector.

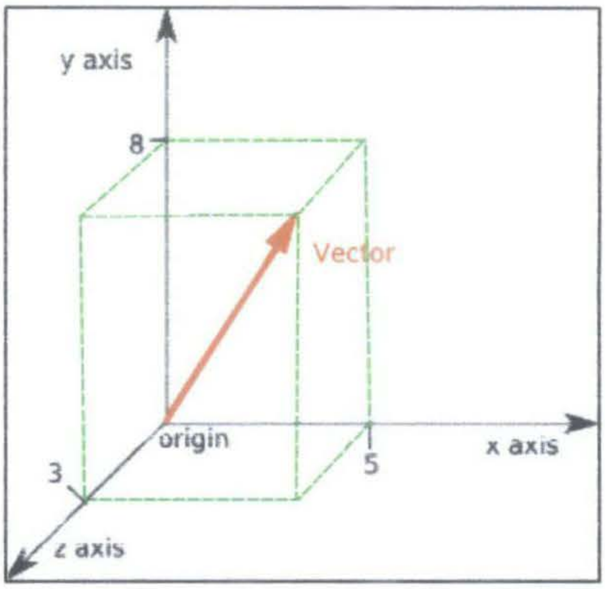


Figure 11: Example of vector (5, 8, 3)

The ball vector is also used to create a spinning ball effects. This is simply done by performing rotation transformation to the ball according to the vector components.

The next physics principle applied to the ball is the gravity. The gravity is the force that 'pulls' the ball down the y axis. To be exact, the gravity force is also can be understood as an acceleration that pulls objects to the y axis. Therefore, gravity is actually acceleration along the -y axis. In this game, gravity force was applied to the ball by subtracting the y component of the ball vector with the gravity constant every frame. This will cause the ball to have a constant negative acceleration along the y axis that 'drags' the ball down, hence, simulating the effects of gravity force.

Another important point of the ball physics is about making the ball bounce realistically. This can be done by using the vector reflection technique. The concept of vector for two dimensions plane can be seen in the figure below.

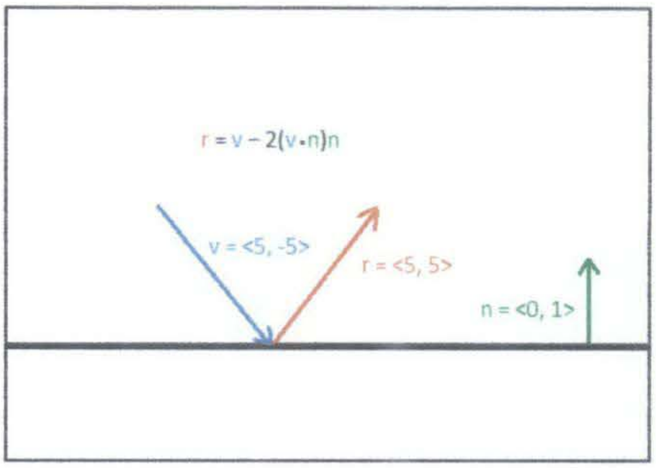


Figure 12: Vector reflection

The original ball vector before the collision (v) and the normal vector of the colliding object surface (n) are used to calculate the reflected vector (r). The formula to find r is r = v – 2 * (v dot n) * n. The 'dot' denotes the dot product operator that calculated the dot product of the two vectors. From this formula, the resulting vector (r) is the new vector for the ball after the collision. Notice that this formula is for 2D plane. For 3D plane, some modification must be done, but it is basically using the same concept.

This technique however will create a perfect bouncing effect. This is not realistic, because perfect bounce (reflection) will never happens in the real world. In real world, a sum of the ball potential energy will lost during the collision/bounce according to the object bounciness (coefficient of restitution). Additionally, the direction of reflection does not exactly follow the formula – because of uneven surface of the objects. For these reasons, all objects in the game has bounciness factor. For instance, if an object with bounciness factor of 0.5 will reduce the ball vector (speed) by half, if the ball collides that object. In addition, some randomness factors are also added to perturb the ball vector after reflection to ensure the ball vector (direction) does not perfectly follow the result of the formula.

4.3 Game Control

4.3.1 Keyboard

The keyboard is the main user input device for this game. With the help of GLUT API, the keyboard can be easily programmed as a controller for this game – by using the glutKeyboardFunc and glutSpecialFunc functions. Therefore, the game can be controlled by using the keyboard.

Below is the table of keyboard key assignments that has been defined for this game:

Key	Action
Up direction	Move player up
Down direction	Move player down
Left direction	Move player left
Right direction	Move player right
A	Change player selection
S	Player perform Instep Kick
D	Player perform Bicycle kick
C	Change camera view
[	Decrease gravity
]	Increase gravity
9	Decrease game speed
0	Increase game speed
R	Reset game

Table 2: Keyboard key assignments

The player can be move around the court by using the direction keys (Up arrow, down arrow, left arrow, right arrow keys) on the keyboard. This is a standard way of controlling a player movement in most games. However, the player movements are limited to their court area only – the player is not allowed to move outside their court.

There are two types of kick that the player can perform. The instep kick and the bicycle kick. Those kicks can be performed by using 'S' key and 'D' key respectively. Instep kick is used to receive and control the ball, while bicycle kick is used to send the ball to the opponent side. The 'S' and 'D' keys are chosen because it has been a tradition for most of sports games to use 'S' for passing and 'D' for shooting.

The user can also change the camera view by using the 'C' key. It will toggle the viewing modes either using Orthographic view or Perspective view. 'C' key was chose because it is an initial for camera which makes it easily remembered.

4.3.2 Player Selection

There are three players for each team. The user can only control one of the team. At a time, only one player can be selected and controlled. The player selection can be cycled by using 'A' key. This is a standard way of controlling player in sport games that consist of multiple players per team. The selected player is indicated with a marker on the player.

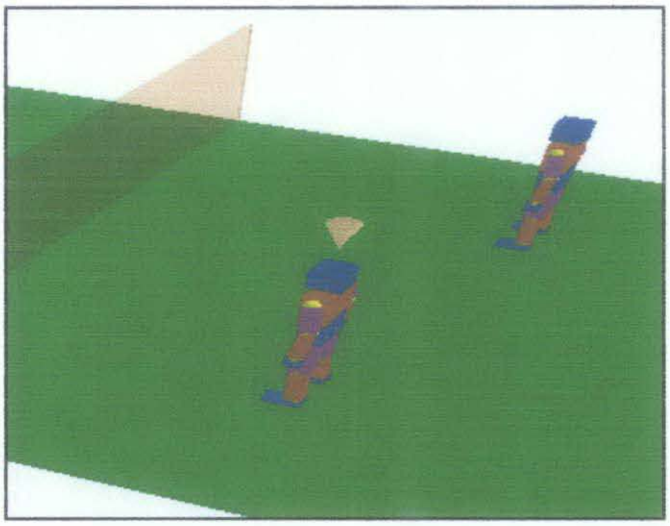


Figure 13: Cursor (pink cone) pointing at the current selected player

4.3.3 Shot Power & Direction

For bicycle kick the power and direction of the shot can be controlled by the user. The power of the shot depends on the duration of the 'D' is pressed. The longer the button is pressed, the more powerful the shot. A bar gauge will appear above the player cursor that represents the show power. If the bar gauge is full, it means the player will hit shot ball with the maximum power. However, the more power is put on the shot, the less accurate the shot will be. The shot direction also can be adjusted by holding the 'Up arrow' or 'Down arrow' button depending on where the user wants to place that shot.

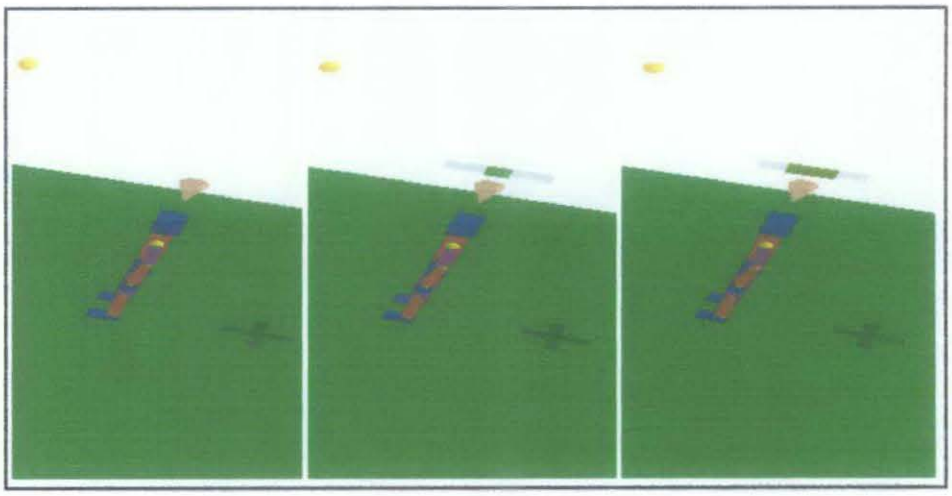


Figure 14: Shot power gauge

CHAPTER 5

5. CONCLUSION AND RECOMMENDATION

This project has accomplished all of its main objectives and a game prototype that simulates the playing of sepak takraw sport in 3D has been developed. The details of the projects results can be seen on the previous Results and Discussion chapter. During this project development, several computer graphics has been explored and used to develop the 3D sepak takraw game prototype. In addition, there are also a number of game physics techniques that were studied and applied in the prototype to enhance the gameplay realism. Although there are some of the game features were dropped during the project development because of time constrains, the game is still playable and able to demonstrate the simulation of sepak takraw play in 3D computer graphics.

5.1 Future Work and Recommendation

For the future work, there is a lot of room of improvements for this project. Since the project only focused on the graphics and physics aspects, there are many other aspects of the game that can be further improved or implemented to enhance the game. Below are some of the aspects that are recommended to be improved or implemented in the future project:

5.1.1 Graphics

The overall game graphics can be significantly improved by implementing the texture. As we can see there is no texture applied in this game. The graphics of the game can also be improved by adding environment objects, i.e. the background, the sky, the audiences, the referee or linesmen. The player animation can also be improved by using a motion capture device to capture motion data from real sepak takraw player. Apart from that, a more sophisticated interpolation technique – such as cosine interpolation or cubic interpolation – can be used for interpolating the player motion instead of using linear interpolation. This will result in more realistic and smoother player animation.

5.1.2 Game Physics

There are many more game physics techniques that can be implemented to make the game more realistic. For instance, the ragdoll physics technique can be implemented to create effects for the player or the net. The current prototype physics engine also does not take friction into account. Calculating friction can be useful in creating spinning effects for the ball and improve the ball physics.

5.1.3 Sound

In the beginning, the sound is part of the features of the game prototype. Unfortunately, due to time constrains, it has been dropped from this project. Implementing the game sounds will definitely improve the overall gaming experience for the user. The examples of sounds that can be implemented are the ball sound effects, the players sound, and the referee voices, and the background music for the menu.

5.1.4 Gameplay

Under the gameplay part, there are also a lot of features that can be added. For instance, more player actions can be added, such as outside kick, front foot kick, knee bump, and header. The artificial intelligence (AI) engine that controls the opponent team also must be improved to make the game more exciting and challenging. The currently implemented AI is not very smart and somewhat predictable. Moreover, there is also no game difficulty setting that allows the user to control the difficulty level. This feature also can be implemented in the future work.

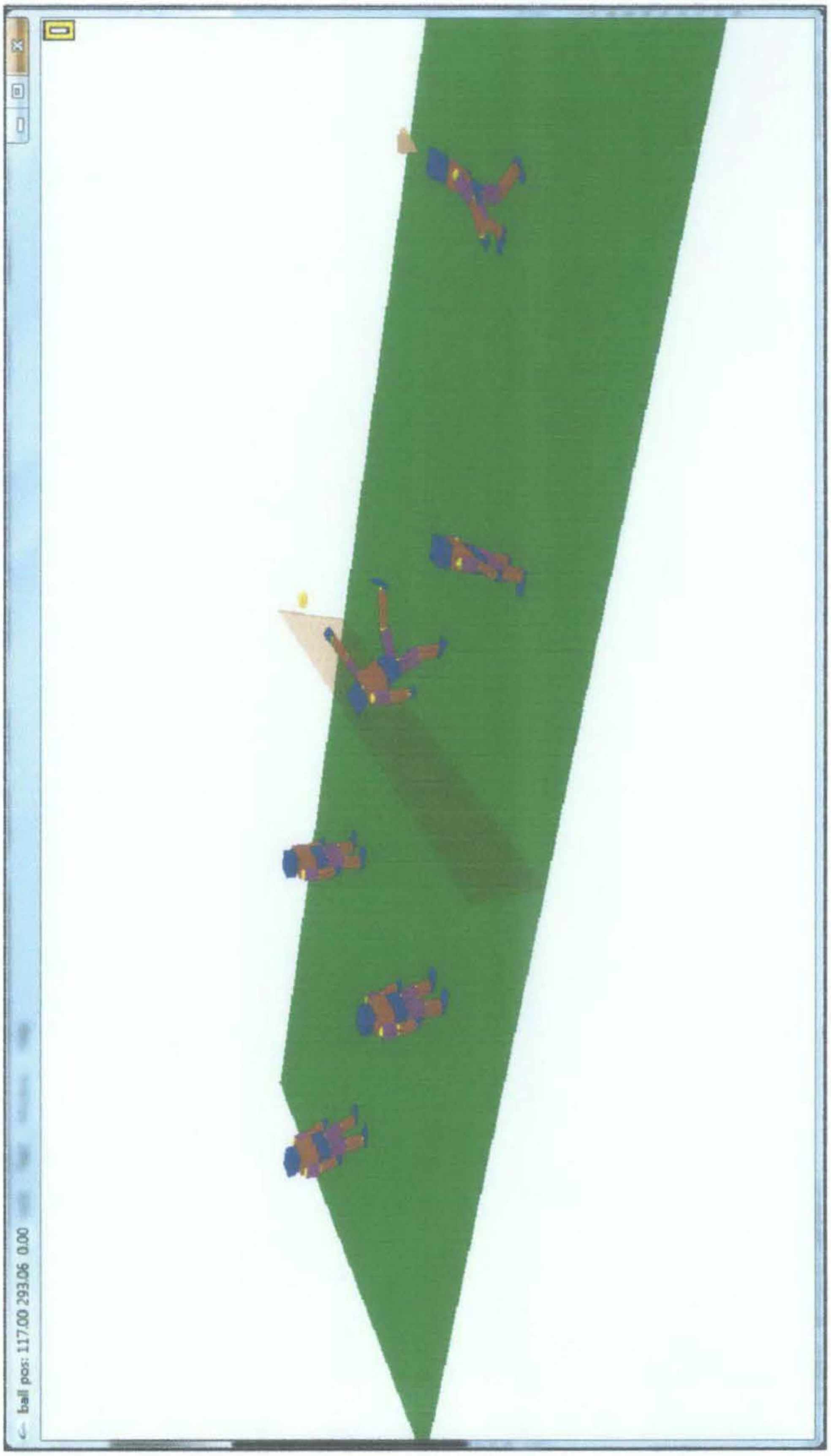
Several game modes also can be added to add more variation to the game like tournament mode, friendly mode, training mode, or even multiplayer mode. Multiplayer mode is also another important feature for most of modern computer games. The game will be more interesting if can be play by multiple players whereby the two or more users can play together as a team or compete against each others. It can be accomplished by allowing multiple users playing on the same computer with different input devices i.e. keyboard and joysticks, or allowing the game to be played by multiple users over the network.

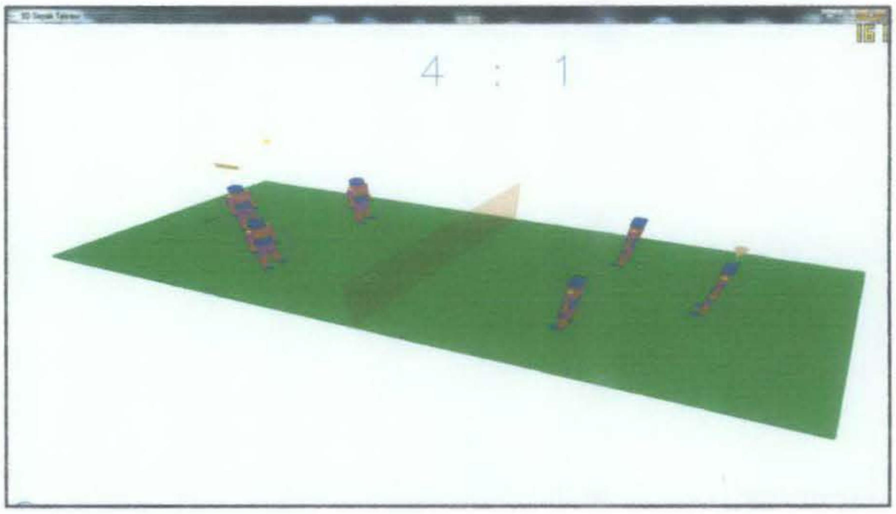
REFERENCES

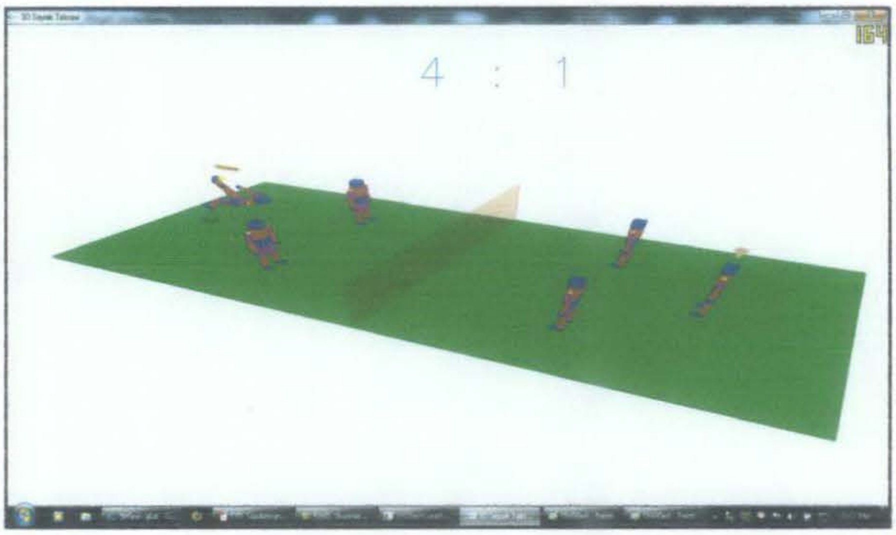
1. Andy Goergen 2010, Nintendo World Report, Review on Deca Sports DS, http://www.nintendoworldreport.com/review/22955
2. Justin Vonderach, "DECA SPORTS DS - History of Sepak Takraw - Sepak Takraw: One of the World's Biggest Extreme Competitive Sports!", http://hudsonentertainment.com/user/feature.php?f=DECA_SPORTS_DS_-_History_of_Sepak_Takraw&feature_id=%A0%A1%A4%A2%9B%A7
3. Szilard Kiss 2002, Computer Animation for Articulated 3D Characters
4. Humanoid Animation Working Group, Web3D Consortium 1999, H-Anim: Specification for a Standard VRML Humanoid, version 1.1. http://www.hanim.org/Specications/H-Anim1.1/
5. International Sepak Takraw Federation (ISTAF), 2004, Sepak Takraw: Laws of the Game
6. Helmut Knaust, S.O.S. MATHematics, Application: A Bouncing Ball http://www.sosmath.com/calculus/geoser/bounce/bounce.html
7. Collin Potts 1998, "Software Process Models" in Introduction to Software Engineering slides
8. Herbert Schildt, 1998, "*C++ The Complete Reference Third Edition*" Osborne McGraw-Hill
9. Computer graphics, http://en.wikipedia.org/wiki/Computer_graphics
10. Alessandro Re, "OpenGL: Tutorials", Basic Bones System, http://gpwiki.org/index.php/OpenGL:Tutorials:Basic_Bones_System
11. Iterative and incremental development, http://en.wikipedia.org/wiki/Iterative_and_incremental_development
12. Michael Hurst, 2005, homepage of English Sepak Takraw Association website, http://sepak-takraw.co.uk/
13. Simon Green, 21st June 2006, "NVIDIA OpenGL Update" Presentation Slide, NVX_instanced_arrays OpenGL extension, http://developer.nvidia.com/object/opengl-nvidia-extensions-gdc-2006.html

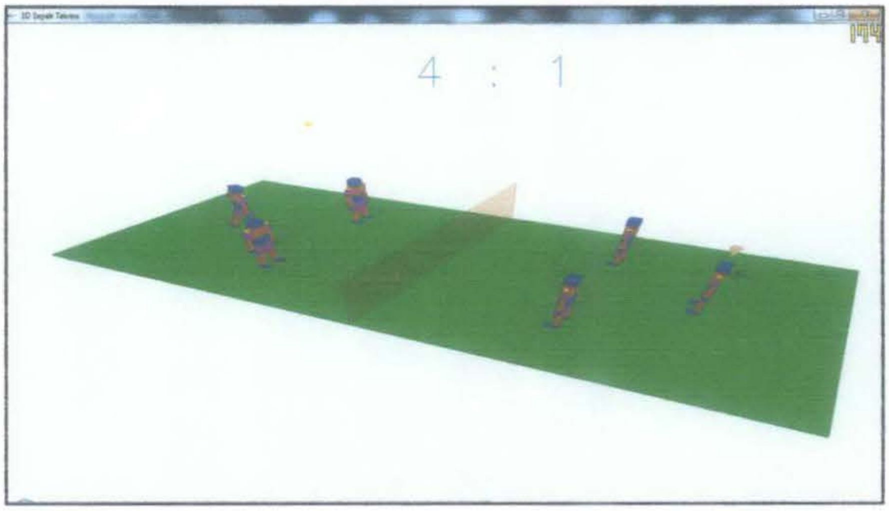
APPENDICES

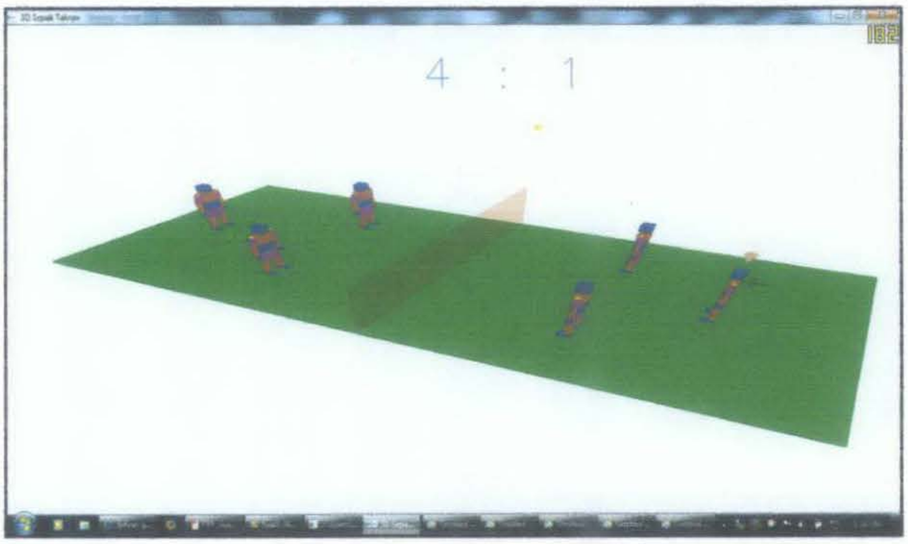
APPENDIX I – Game Screenshots


ball pos: 117.00 293.06 0.00


4 : 1


4 : 1


3D Sepak Takraw
174
4 : 1


3D Sepak Takraw
182
4 : 1

APPENDIX II – Human Figure Pose Data

```
#ifndef GAME_DATA_H
#define GAME_DATA_H

typedef HFigurePoses Action;

namespace GD{
        Action IDLE;
        Action BKICK;
        Action MOVE1;
        Action MOVE2;
        Action IKICK;

        bool isLoaded = false;

        void loadPosesData(){
                if(isLoaded)
                        return;

                HFigurePose reset;
                reset.rShoulder.set(4,-5,-5);
                reset.lShoulder.set(4,5,5);

                reset.rElbow.set(-10,0,0);
                reset.lElbow.set(-10,0,0);

                reset.rHip.set(-2,-3,-7);
                reset.lHip.set(-2,3,7);

                reset.rKnee.set(3,0,0);
                reset.lKnee.set(3,0,0);

                reset.rAnkle.set(-2,0,5);
                reset.lAnkle.set(-2,0,-5);

                reset.rootS.set(0,2,0);

////////////////////////////////////////////////////////////////////////////

                IDLE.keyPoseAt(0).set(reset);

////////////////////////////////////////////////////////////////////////////

                BKICK.keyPoseAt(0).set(reset);

                BKICK.keyPoseAt(35).lKnee.set(85,0,0);
                BKICK.keyPoseAt(35).rKnee.set(85,0,0);

                BKICK.keyPoseAt(35).lAnkle.set(-10,0,0);
                BKICK.keyPoseAt(35).rAnkle.set(-10,0,0);

                BKICK.keyPoseAt(35).lHip.set(-30,-10, 10);
                BKICK.keyPoseAt(35).rHip.set(-30,10,-10);

                BKICK.keyPoseAt(35).neck.set(-25,0,0);
                BKICK.keyPoseAt(35).spine.set(5,0,0);

                BKICK.keyPoseAt(35).lShoulder.set(-30,0,10);
                BKICK.keyPoseAt(35).rShoulder.set(5,0,-10);

                BKICK.keyPoseAt(35).lElbow.set(-60,30,0);
                BKICK.keyPoseAt(35).rElbow.set(-30,-10,0);

                BKICK.keyPoseAt(35).rootS.set(0,15,0);
                BKICK.keyPoseAt(35).rootJ.set(0,180,0);

                BKICK.keyPoseAt(50).lKnee.set(20,0,0);
                BKICK.keyPoseAt(50).rKnee.set(10,0,0);

                BKICK.keyPoseAt(50).lAnkle.set(30,0,0);
                BKICK.keyPoseAt(50).rAnkle.set(60,0,0);
BKICK.keyPoseAt(50).lHip.set(-110,-10, 10);
BKICK.keyPoseAt(50).rHip.set(-5,10,-10);

BKICK.keyPoseAt(50).neck.set(-20,0,0);
BKICK.keyPoseAt(50).spine.set(0,0,0);

BKICK.keyPoseAt(50).lShoulder.set(-120,0,10);
BKICK.keyPoseAt(50).rShoulder.set(40,0,-10);

BKICK.keyPoseAt(50).lElbow.set(0,30,0);
BKICK.keyPoseAt(50).rElbow.set(0,-10,0);

BKICK.keyPoseAt(50).rootS.set(0,-5,0);
BKICK.keyPoseAt(50).rootJ.set(20,180,0);

BKICK.keyPoseAt(65).lKnee.set(40,0,0);
BKICK.keyPoseAt(65).rKnee.set(20,0,0);

BKICK.keyPoseAt(65).lAnkle.set(0,0,0);
BKICK.keyPoseAt(65).rAnkle.set(20,0,0);

BKICK.keyPoseAt(65).lHip.set(-90,0,0);
BKICK.keyPoseAt(65).rHip.set(20,0,0);

BKICK.keyPoseAt(65).neck.set(-20,0,0);
BKICK.keyPoseAt(65).spine.set(-5,0,0);

BKICK.keyPoseAt(65).lShoulder.set(-130,0,10);
BKICK.keyPoseAt(65).rShoulder.set(-40,0,-10);

BKICK.keyPoseAt(65).lElbow.set(-10,0,0);
BKICK.keyPoseAt(65).rElbow.set(-10,0,0);

BKICK.keyPoseAt(65).rootS.set(0,-20,0);
BKICK.keyPoseAt(65).rootJ.set(40,180,0);

BKICK.keyPoseAt(85).lKnee.set(55,0,0);
BKICK.keyPoseAt(85).rKnee.set(10,0,0);

BKICK.keyPoseAt(85).lAnkle.set(0,0,0);
BKICK.keyPoseAt(85).rAnkle.set(40,0,0);

BKICK.keyPoseAt(85).lHip.set(0,0,0);
BKICK.keyPoseAt(85).rHip.set(-100,0,0);

BKICK.keyPoseAt(85).neck.set(-10,0,0);
BKICK.keyPoseAt(85).spine.set(10,0,0);

BKICK.keyPoseAt(85).lShoulder.set(-70,0,10);
BKICK.keyPoseAt(85).rShoulder.set(-30,0,-10);

BKICK.keyPoseAt(85).lElbow.set(-10,0,0);
BKICK.keyPoseAt(85).rElbow.set(-10,0,0);

BKICK.keyPoseAt(85).rootS.set(0,-15,0);
BKICK.keyPoseAt(85).rootJ.set(80,180,0);

BKICK.keyPoseAt(95).lKnee.set(55,0,0);
BKICK.keyPoseAt(95).rKnee.set(10,0,0);

BKICK.keyPoseAt(95).lAnkle.set(0,0,0);
BKICK.keyPoseAt(95).rAnkle.set(40,0,0);

BKICK.keyPoseAt(95).lHip.set(0,0,0);
BKICK.keyPoseAt(95).rHip.set(-100,0,0);

BKICK.keyPoseAt(95).neck.set(0,0,0);
BKICK.keyPoseAt(95).spine.set(10,0,0);

BKICK.keyPoseAt(95).lShoulder.set(-70,0,10);
BKICK.keyPoseAt(95).rShoulder.set(-30,0,-10);

BKICK.keyPoseAt(95).lElbow.set(-10,0,0);
BKICK.keyPoseAt(95).rElbow.set(-10,0,0);

BKICK.keyPoseAt(95).rootS.set(0,-10,0);
BKICK.keyPoseAt(95).rootJ.set(120,180,0);
```

```
                BKICK.keyPoseAt(125).lKnee.set(65,0,0);
                BKICK.keyPoseAt(125).rKnee.set(65,0,0);

                BKICK.keyPoseAt(125).lAnkle.set(-10,0,0);
                BKICK.keyPoseAt(125).rAnkle.set(-10,0,0);

                BKICK.keyPoseAt(125).lHip.set(-90,0,0);
                BKICK.keyPoseAt(125).rHip.set(-110,0,0);

                BKICK.keyPoseAt(125).neck.set(15,0,0);
                BKICK.keyPoseAt(125).spine.set(10,0,0);

                BKICK.keyPoseAt(125).lShoulder.set(-90,0,10);
                BKICK.keyPoseAt(125).rShoulder.set(-80,0,-10);

                BKICK.keyPoseAt(125).lElbow.set(-20,0,0);
                BKICK.keyPoseAt(125).rElbow.set(-20,0,0);

                BKICK.keyPoseAt(125).rootS.set(0,20,0);

                BKICK.keyPoseAt(149).rootJ.set(358,160,0);
                BKICK.keyPoseAt(150).rShoulder.set(22,-10,-10);
                BKICK.keyPoseAt(150).lShoulder.set(-22,10,10);

                BKICK.keyPoseAt(150).rElbow.set(-22,0,0);
                BKICK.keyPoseAt(150).lElbow.set(-22,0,0);

                BKICK.keyPoseAt(150).rHip.set(-25,-10,-10);
                BKICK.keyPoseAt(150).lHip.set(20,10,10);

                BKICK.keyPoseAt(150).rKnee.set(25,0,0);
                BKICK.keyPoseAt(150).lKnee.set(35,0,0);

                BKICK.keyPoseAt(150).rAnkle.set(10,0,0);
                BKICK.keyPoseAt(150).lAnkle.set(15,0,0);

                BKICK.keyPoseAt(150).rootS.set(0,5,0);
                BKICK.keyPoseAt(150).rootJ.set(0,150,0);

                BKICK.keyPoseAt(180).set(reset);

                BKICK.interpolatePoses();

//////////////////////////////////////////////////////////////////////////////

                const int move1poseCount = 40;

                MOVE1.keyPoseAt(0).set(reset);

                MOVE1.keyPoseAt(move1poseCount/2).rShoulder.set(8,-10,-10);
                MOVE1.keyPoseAt(move1poseCount/2).lShoulder.set(8,10,10);

                MOVE1.keyPoseAt(move1poseCount/2).rElbow.set(-22,0,0);
                MOVE1.keyPoseAt(move1poseCount/2).lElbow.set(-22,0,0);

                MOVE1.keyPoseAt(move1poseCount/2).rHip.set(-5,-7,-15);
                MOVE1.keyPoseAt(move1poseCount/2).lHip.set(-5,7,15);

                MOVE1.keyPoseAt(move1poseCount/2).rKnee.set(15,0,0);
                MOVE1.keyPoseAt(move1poseCount/2).lKnee.set(15,0,0);

                MOVE1.keyPoseAt(move1poseCount/2).rAnkle.set(-5,0,10);
                MOVE1.keyPoseAt(move1poseCount/2).lAnkle.set(-5,0,-10);

                MOVE1.keyPoseAt(move1poseCount/2).rootS.set(0,5,0);

                MOVE1.keyPoseAt(move1poseCount).set(reset);

                MOVE1.interpolatePoses();

//////////////////////////////////////////////////////////////////////////////

                const int move2poseCount = 40;

                MOVE2.keyPoseAt(0).set(reset);
                MOVE2.keyPoseAt(move2poseCount/2).rShoulder.set(22,-10,-10);
                MOVE2.keyPoseAt(move2poseCount/2).lShoulder.set(-22,10,10);

                MOVE2.keyPoseAt(move2poseCount/2).rElbow.set(-22,0,0);
                MOVE2.keyPoseAt(move2poseCount/2).lElbow.set(-22,0,0);

                MOVE2.keyPoseAt(move2poseCount/2).rHip.set(-25,-2,-2);
                MOVE2.keyPoseAt(move2poseCount/2).lHip.set(20,2,2);

                MOVE2.keyPoseAt(move2poseCount/2).rKnee.set(25,0,0);
                MOVE2.keyPoseAt(move2poseCount/2).lKnee.set(35,0,0);

                MOVE2.keyPoseAt(move2poseCount/2).rAnkle.set(10,0,0);
                MOVE2.keyPoseAt(move2poseCount/2).lAnkle.set(15,0,0);

                MOVE2.keyPoseAt(move2poseCount/2).rootS.set(0,5,0);

                MOVE2.keyPoseAt(move2poseCount).set(reset);

                MOVE2.interpolatePoses();

////////////////////////////////////////////////////////////////////////////////

                IKICK.keyPoseAt(0).set(reset);

                IKICK.keyPoseAt(30).rShoulder.set(-10,-10,-10);
                IKICK.keyPoseAt(30).lShoulder.set(-10,10,10);

                IKICK.keyPoseAt(30).rElbow.set(-22,0,0);
                IKICK.keyPoseAt(30).lElbow.set(-22,0,0);

                IKICK.keyPoseAt(30).rHip.set(-110,-90,-40);
                IKICK.keyPoseAt(30).lHip.set(-20,5,5);

                IKICK.keyPoseAt(30).rKnee.set(75,0,0);
                IKICK.keyPoseAt(30).lKnee.set(35,0,0);

                IKICK.keyPoseAt(30).lAnkle.set(-10,0,0);

                IKICK.keyPoseAt(30).spine.set(10,0,0);

                IKICK.keyPoseAt(30).neck.set(8,0,0);

                IKICK.keyPoseAt(30).rootS.set(0,5,0);
                IKICK.keyPoseAt(30).rootJ.set(-8,0,0);

                IKICK.keyPoseAt(60).set(reset);


                IKICK.interpolatePoses();

////////////////////////////////////////////////////////////////////////////////

                isLoaded = true;
        }
}

#endif
```