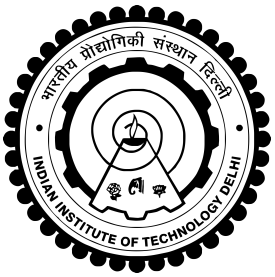
IIT Delhi

COP 290 : Design Practices in Computer Science

Assignment 3 : Haxball

Final Design Document With Changes

Anmol Sood - 2013MT60587
Kartikay Garg - 2013MT60600
Rahul Kumar Rathore - 2013MT60610



0.1 Abstract

The objective of this project is to design a real-time multiplayer game using the concept of P2P(Peer to Peer) architecture. The project also intends to handle any disconnections which might happen during the game and keep the game synchronized at all clients. This game also includes a basic AI module that allows the user to play with the computer without the requirement of any other player.
Gameplay has already been included in the presentation.

0.2 Introduction

Haxball is a very famous game based on multiplayer design and Peer to Peer synchronization. In this project, we will try to recreate this game in its true essence and will also try to add additional features such as **AI Module**, **Chat Room** which are not included in the original game. These features and other design steps and procedures are described in the following sections.

0.3 Setup and Software Used

Environment : **Ubuntu 14.04**
Language : **C++**
GUI : **QT**, **OpenGL** , **SFML**
Communication : **UDP Sockets**
UDP Layer(Networking) : **ENet**
Debugger : **gdb**
Documentation : **LaTeX**

0.4 Overall Design

The game is based on P2P connection among the peers using UDP sockets without the use of central server to maintain game state. Data packets will be exchanged among different players directly, will be processed at individual processors and the results will be displayed on the screen. The GUI of the game is designed using QT and OpenGL where as the Physics and Gameplay is implemented using vector algebra and other basic laws of Physics. The game also includes an AI module of a basic level(strategy based module). We handle various kinds of disconnection that might arise in the network. The game starts in a lobby where the host is able to divide the players into teams and select the various available game options.

0.5 Sub Components

1. **Physics**
2. **Networking**
3. **GUI**
4. **AI**

0.5.1 Physics

The physics of the game is simple and efficient. It completely relies on the basic concepts of Conservation of momentum and Coefficient of Restitution and vector algebra. Functionalities provided to the players by the physics of the game as follows :

- Players can carry the ball along with them in the direction of their movement at the speed of their motion.
- Player can kick the ball with a high intensity but the direction of the kick is decided by the alignment of the centers of the ball and the player bot.
- The ball will be reflected upon hitting the boundaries of the stadium. Goal is detected when complete ball crosses goal line.

0.5.2 Networking

UDP Ports with the help of ENet Library have been used to implement a P2P architecture. Since UDP connections are unordered, unreliable and connectionles, we have designed proper mechanisms to handle these issues and also to detect and handle disconnections.

The game starts with the host making a lobby. The concept of connection that we have introduced is the continuous sending of packets between the two connected peers. To initiate a connection , the peer first sends a packet to the host containing its current information. Once this packet is received, the host sends a packet describing the current lobby state. Also, the information of the new peer is broadcasted to all other peers so that connections can be formed between each pair of peers in the network. Any changes which the host makes during the lobby phase are broadcasted to all other players. Once the host starts the game, the host behaves as a normal peer.

The game is based on a lockstep model, i.e we wait for packets containing relevant gamestate from all other peers at each peer. If this packet is not received, we try to resend it uptil a maximum of 32 times, failing which we detect a disconnection and handle it accordingly. Once , we receive the information for all peers , we draw the next frame and compute the new gamestate at each peer. Each peer sends only the data which is relevant to

its own player to all other peers. Any game events(like a goal scored,end of game) are also broadcasted to all peers on a separate channel to maintain synchronization.
If a disconnection is detected (i.e if no packets are exchanged between two peers until the timeout period), we handle the disconnection based on whether the host is involved or not,i.e the host is a preferred peer for disconnections. If the connection between the host and a peer breaks, then the peer is considered to be disconnected from the game, i.e it will not move, all other players continue playing the game. If the disconnection is between two non-host peers, then both players are consider disconnected from the game until only one of them is connected to the host in which case only the other player is considered disconnected.
The network has an event based model , i.e any receipt or sending of packet is an event which is handled accordingly. We continuously pool for events and they are handled sequentially. This is done with the help of Enet library.

0.5.3 GUI

GUI of the game can be majorly divided into 2 components. These include **Game Lobby** and **Stadium** of the game.

- **Lobby**
  We are using QT for creating game lobby . Each player joins the host of the game. Each player has to provide a nick name. They will also have to choose a side between two teams by clicking a **JOIN TEAM** button. Then host starts the game. Players can also chat with each other using the chat box provided at the bottom left of the screen.
- **Stadium**
  Opengl will be used for this purpose. The stadium is created using lines and polygons in OpenGL. The complete graphics of the stadium and the players is done using OpenGL. The player bot is free to move in the whole screen whereas the ball is completely constrained within the walls of the stadium from where it is reflected back into the stadium. Other additional features are mentioned in the sections to follow.

0.5.4 AI

The AI framework of the game is fairly simple. The user will only be able to play against a single bot player and can adjust the hardness level of the game as she wishes. AI of the game is implemented as follows :

- The AI will implement a number of predefined strategies based upon the outcome of a random event. Changing the probabilities of the

occurrence of these strategies changes the hardness level of the game. Different strategies implemented will be as follows :

1. Bot will move randomly move about in the field. This strategy has minimum probability.
2. Bot will try to align itself in a direction around the ball so that a direct kick results in a goal.
3. Bot will try to align itself in a direction around the ball so that a diagonal kick results in a goal.
4. Bot will try to block the paths of other players.

- And various other strategies are adopted and calculations are done in each frame so that the optimal move is decided.
- Increasing the probabilities of the occurrence of the strategies 2 and 3 will increase the hardness level of the game where as reverse happens when we increase the probability of the occurrence of strategy 1.
- We have kept three different hardness levels for the user to choose from.

0.6 Additional Features

- The duration of the game is monitored and controlled by digital clock present on the stadium screen.
- Players can chat with each other in the game lobby before starting the game.
- AI has three different hardness levels namely Easy, Medium and Hard levels that the user can select before starting the game.
- AT bot can used to fill the team if the host wishes to.
- Goal posts also participate in the game. They reflect the ball as it strikes them but allows the players to pass through.