Najwa Harif
Sarah Mauret
Lucas Chaulan
Elliot Hallais
Hugo Blech

Final Programming Project

ECE 6122

INTRODUCTION

This report details the final project for the class ECE 6122.

We are five ECE exchange students from France, and the project we decided to do is a Beer pong game in C++.

PROJECT

- The choices we made

For this project, we chose to use Urho3D engine, since its official documentation is quite complete compared to other engines. Some of us are quite new to programming projects containing graphics content, so the use of this engine was a good start.

We decided the best option is to represent the project in 2D graphics, but keep all the backbone and trajectory calculations in 3D. We did all the trajectory equations by ourselves so the conversion between a movement in the screen and a movement in the game table reference was challenging by moments. The trajectories and constants were modified afterwards to fit the reality.

As we wanted to explore the possibility of having a 3D graphics solution, we decided to assign one team member to this task, while the rest of the team was focused on developing a 2D solution that would be viable.

- The game course

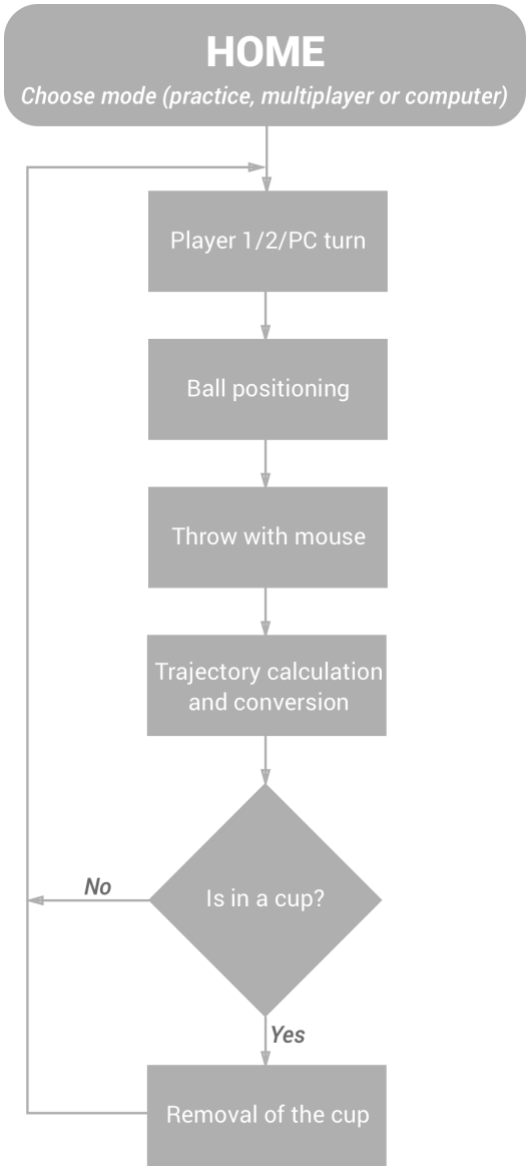
The project is a 2D beer pong game with 3 game modes:

- Practice: the user plays alone for practicing
- Multi: two users compete with each other
- Computer: the user plays against the computer

The course of the game for the multiplayer mode is as follow:

1. Player 1 clicks anywhere on the screen to set the initial position of the ball. He has to click a second time on the ball to drag and drop it in the air. The throw of the ball starts when the player release the mouse.
2. The ball is thrown according to the player's input and is animated to follow the trajectory computed by the code.
3. Depending on if the ball is scored in a cup, an animation can appear. The second player can play as soon as the ball stop moving.
4. Player 2 clicks anywhere on the screen to set its initial ball position and the view is changed to show the table from his point of view. He drags and drops the ball to launch it.
5. The game continues the same way until one of the players scores all his cups.

The game in computer mode has the same course of action except the player 2 is the computer and the throw of the computer is generated in the code with a random factor. In the solo mode, the user keeps playing until he wins, there is no change of view, only the cups are updated.



Above is the functional graph of our game.

CORE CLASSES OVERVIEW

Following are the main classes used as backbone of the game, only important attributes and functions are mentioned here.

- Class Player:

The class Player is the class that define a player (computer or user). It is in this class in which the trajectory of the ball is computed through the throBall function:

For the user:

throwBall(alpha: double, beta: double, h: double, v0: double, startX: double, startY: int, &cupScored: int) : vector<Vec3<int>> The function take in input the characteristics of the throw such as the angle the throw make with the table and the rotation angle of the player, the speed and the position of the ball at the beginning of the throw. It returns a vector containing all the position taken by the ball during its trajectory in the air. The “cupScored” parameter enable us to stored the cup scored by the player when he scores.

For the computer:

throwBall(&cupScored: int) : vector<Vec3<int>> The function that throws the ball for the computer. It generates random throwing characteristics and calls the throwBall function used for the user.

The throwBall function returns the trajectory of the ball in the backbone point of vue: that means all the coordinates are created in an absolute referential and stored in centimeters. We had to create an other function to convert those values in coordinates that can be plotted by urho3d (in pixels).

get_xzSize_graphics(&ballTrajectory: vector<Vec3<int>>, &graphicsTrajectory: vector<Ven3<int>>, &cupScored: int): void The function that converts all the informations returned by the throwBall function into pixels position that can be understand by urho3d.

- Class Cup:

The class Cup represents the cups in which the player has to score in order to win the game. This is a very simple class that is just used to stored the cups positions on the table and whereas they have been scored already.

Now in the main.cpp we use those classes to define the game.

- Class main:

This is the main class of the project. All the game is run and monitored in this class.

This class includes:

- *Sample* class, which is a framework for all Urho3D samples, and allows to initialize the engine, modify its parameters, set the window, create the console, handles Esc key down to hide Console or exit application.
- *Urho3D/UI* elements, which allow use to create UI elements from the engine, such as buttons or windows.
- *Urho3D/Graphics* to be able to use the graphics class and handle 2D textures.
- More generally the *engine* and its *core events*.

It contains two pages:

- The welcome page contains a background image and a welcome text, and it allows the user to choose the game mode (Practice, multi, computer).
- The board game page contains all the items composing the game. Some elements are changed and are updated during the game, such as the announcement of the score, the number of throws, or the view of the game..

The elements are stored in two vectors:

- *uielem[]*: Stores pointers to all the UIElements - buttons, windows, ball cursor, background images …
- *main[]*: Stores the cups (which are of type *Sprite*)

Those vectors allow us to easily manage the elements, remove them if needed or add them again without re-creating new ones.

Four main events are managed:

- *E_DRAGBEGIN*: Beginning of the dragging movement. When the user clicks on the ball cursor.
- *E_DRAGMOVE*: Update of the position of the ball according to movement of the cursor

- *E_DRAGEND*: End of the dragging movement. In its handler, we calculate the rotation, speed and positions to send to "throwball" function. Once the trajectory is computed, the movement of the ball is triggered.
- *E_UPDATE*: Logic update event, used to move the ball automatically at each time frame according to trajectory vector.

Other events are used to monitor the buttons.

CONTRIBUTIONS

Najwa: Worked on the creation of the game. She created the beerponggame project and linked it to Urho3D library. She worked on the main.cpp class by creating the scene (welcome page and game page), managing the ball cursor movement before and after throwing the ball, displaying the result etc. She also worked with backbone team members on how to integrate the graphics part with the backbone part (coordinates matching, converting and sending the variables of the throw etc.). Overall, Najwa worked on delivering the minimum viable product, containing the main functionalities of the game.

Sarah: Worked with Lucas on defining the architecture of the game and on implementing the trajectory of the ball given the speed, angle and beginning position of the ball. She implemented the game mode multiplayer, allowing two users to play against one another and switching/updating views accordingly. She created the game mode computer with auto generated ball trajectory with some randomness in the throw. On the graphic side, she worked on managing the layering of all scene elements, making the ball disappear behind cups or the table when needed. She added scene elements (splash when ball falls in cup, different cups and background depending on player…). She did bug fixing and testing.

Lucas: Worked principally on the backbone of the game. He created the different classes that compose the game, such as the Table, the Player, the Cup, the Ball. He worked with Sarah on the trajectory of the ball and they realized a function that returns an accurate trajectory that perfectly matches the trajectory would have had in the air. Moreover, Lucas worked on interfacing the backbone trajectory and the graphics one (the coordinates used by Urho3d to plot the window). Lucas kept updating the backbone code to include the new features the team wanted to include to the game and to help the graphics development when needed.

Elliot: Worked mainly on the UI design, using Sketch (design software on macos) to design all the components. He did the custom styles for the buttons and other UI components (textures, cups, splash effect, table, backgrounds) and the integration on the code (positioning, implementing the correct parameters that are used in the Urho framework). He also spent time on the dimensions analysis of the elements to make sure we have a visualization corresponding to what we had in mind (for the perspective and 3D-like design). He worked also on the end of the game (winning conditions, message display, number of throws). Part of team meetings about game physics for integration of backbone calculations to graphical interface (to make sure everybody was on the same page before merging works). He also did some testing and debugging for the game on the later stages.

Hugo: Worked on the 3D implementation of the game. He focused mostly on implementing the game using the physics engine that Urho3D provides. He conceived the 3D scene using some of the templates that Urho3D provides, and designed the components that are present in the scene. He tried as much as possible to implement the game using only the physics engine provided. Some of

this work is put on the representation of the physical model of the environment, and on the simple game mechanics that were put in place.

SETUP

To compile this project with Urho3D dependencies:

- Unzip the given file
- Go to the root of the project
- Execute compilationScript.sh (it will do all the steps needed (create a build directory in include/Urho3D, then compile Urho3D inside that build folder ; then creating a cmake-build-debug folder at root of project and finally compile the project itself into this cmake-build-debug folder).
- To test the first program that is in the build directory, run ./cmake-build-debug/bin/beerpongproject_emptyscene from the root.

Note: Watch out for Urho3D syntax URHO3D_DEFINE_APPLICATION_MAIN(). Because of the CMake utilities Urho3D provide, only one C++ file can contain a main function in the directory at the same time. Otherwise, compilation is impossible.

YOUTUBE LINK

Here is a link of a video simulation of our program in its different game modes.

https://www.youtube.com/watch?v=XnfIaigwSbA

If the link does not work, you can find it here:

https://www.dropbox.com/s/r2xcnzbu38kgche/Capture.mp4?dl=0

GITHUB LINK

We used GitHub platform to share and build our project.

The link is the following : https://github.com/smauret/beerponggame.git

Note: User “Mac” is actually our team member Najwa Harif, we only realized at the end of the project that there was a typo in her email address, making her an “outside” collaborator on GitHub.