CS 480: Computer Graphics
Project 9: Air Hockey

M. Daniel Chavez
Matthew VanCompernolle

Overview:

- Source code provided and compiles
- Board, paddles and puck models are all loaded using the Assimp library
- All objects are textured
- Camera movement is provided
- Mouse control of Player 1 paddle
- Menu Items restart, pause, resume, exit and change puck and paddle options
- Scores are displayed at the top of the screen along with the status of the game.

Extra Credit:

- **Changeable paddle shapes:** Square and Triangle paddles implemented
- **Puck Shapes:** Square and Triangle pucks implemented

Table of Contents:

- **User Manual**
- **Tech Manual**

User Manual:

To run Air Hockey run the following command from the bin directory:

```
./AirHockey
```

How the game works:

- When game is started a virtual coin toss is ran, and decides what side to place the puck.
- There are two players and a puck. Each player moves their paddle to try to hit the puck into the other player's goal. The puck gracefully glides across the surface of the hockey table and bounces off of objects.
- A player scores when the puck goes into the opponent's goal. The player that was scored on receives the puck.
- In order for a player to win they must reach a score of 7.
    - The game pauses when a player wins and the status of the game is set to "PLAYER X WINS" where 'X' is the player that won.
    - User must restart the game after a win from the options menu (right click).
- When a puck is hit out of bounds the hockey gods decide which side to place the puck, which is usually somewhere in the middle.
- Puck and game resets have random animations and label the status of the game as "resetting".

User Controls:

Player 1: The mouse controls the movement of the paddle for player 1.
Player 2: The "WASD" keys control the move of the paddle for player 2 forward, back, left, and right.
1-4 changes the speed of player 2's paddle with 1 being slowest and 4 being fastest.

Camera Movement: Through the use of the arrow keys the user can control the movement of the camera. Camera movement is limited so the mouse controls stay consistent.

- UP ARROW: Zooms the camera in. The camera snaps to bird's eye view when up close.
- DOWN ARROW: Zooms the camera out (limited distance).
- LEFT ARROW: Moves the camera left (limited distance).

- RIGHT ARROW: Moves the camera right (limited distance).

Bringing up menu items:
Using the mouse a player can right click anytime to bring up menu items. Left clicking on a menu item selects it.
These menu items include:

- **Restart** - Resets score, and positions of paddle and puck
- **Pause** - Pauses the game
- **Resume** - Resumes game from paused state
- **Set Puck Shape** - Sets the puck shape (shape options in submenu)
    - *Normal*
    - *Triangle*
    - *Square*
- **Set P1 Paddle Shape** - Sets the paddle shape (provided in submenu)
    - *Normal*
    - *Triangle*
    - *Square*
- **Set P2 Paddle Shape** - Sets the paddle shape (provided in submenu)
    - *Normal*
    - *Triangle*
    - *Square*
- **Exit** - Exits the game

Menu Items can be seen in figure 3 to figure 5.

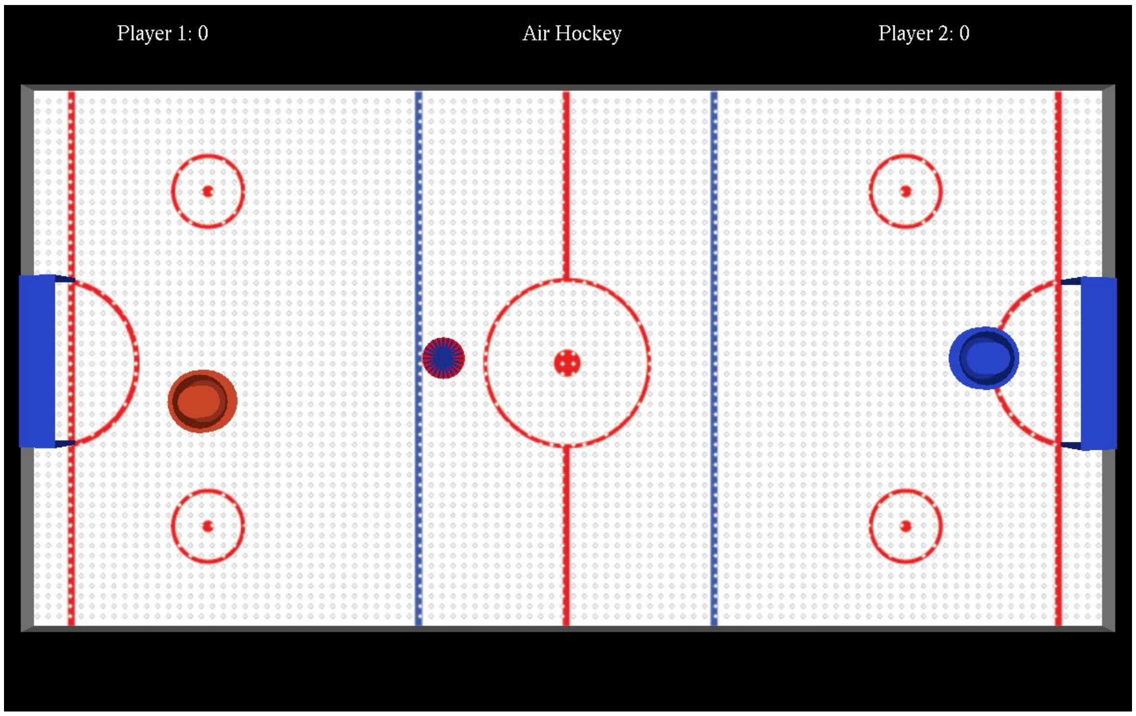

Fig. 1: When the game is started users are presented with a top view of the game. Player 1 is on the left along with their score. Player 2 is on the right along with their score.

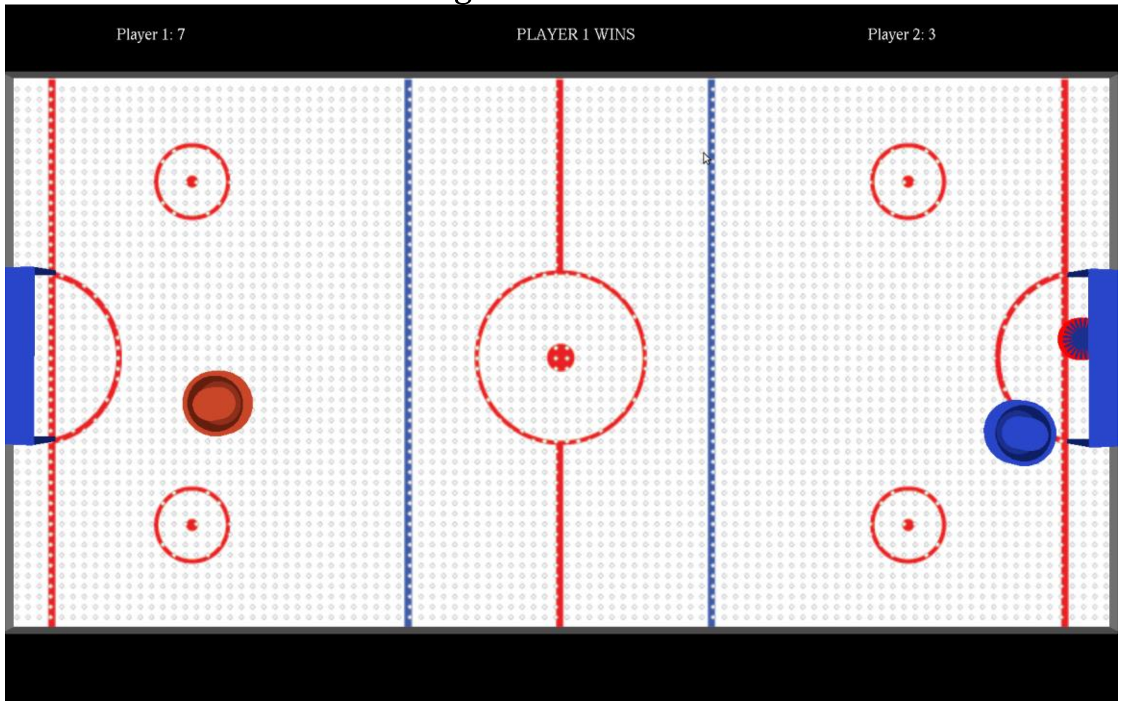

Fig. 2: When a player wins the player who won is displayed at the top of the screen and the final score can be seen until the user restarts the game.

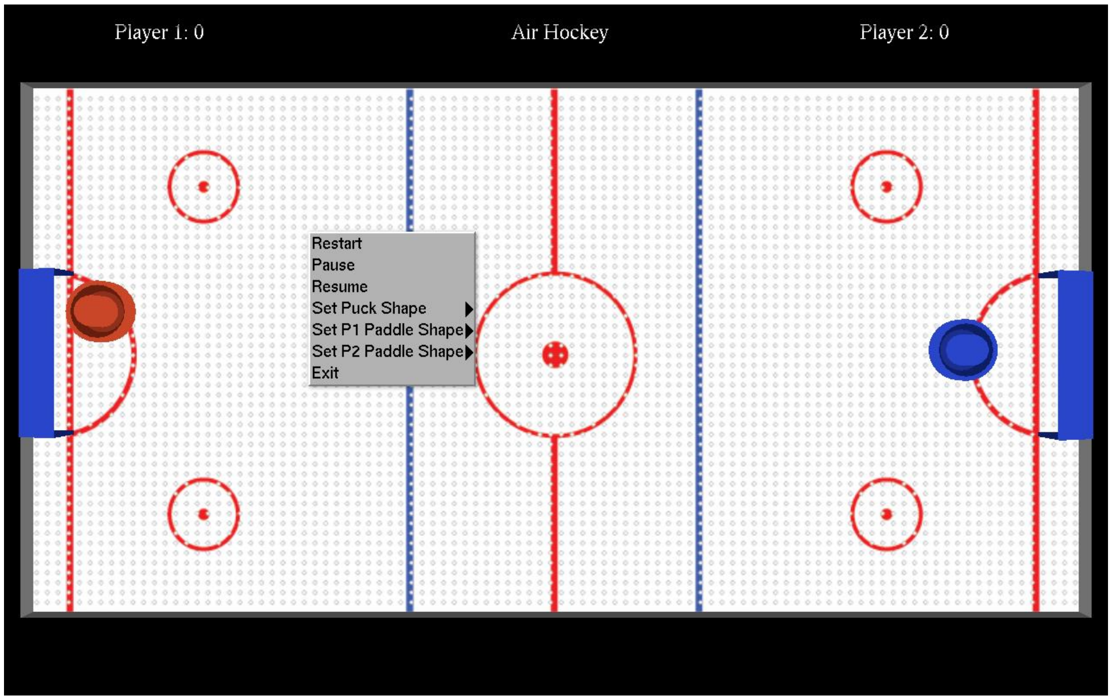

Fig. 3: When a user right clicks the menu is displayed as shown in the image above.

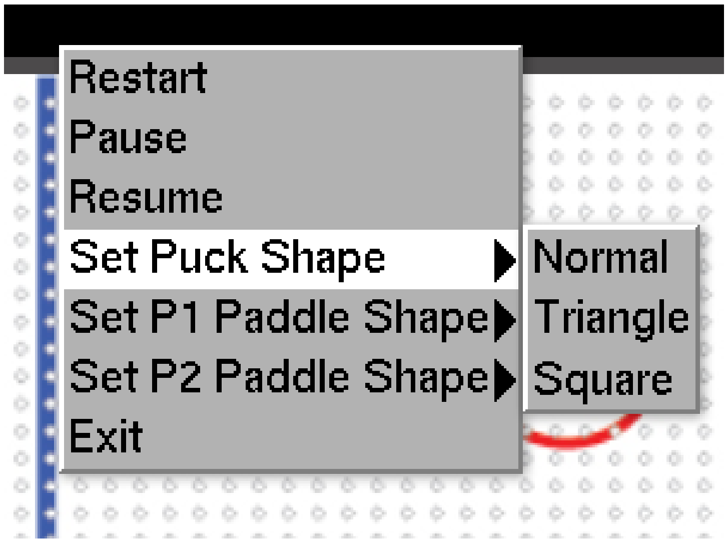

Fig. 4: When set puck shape is selected a submenu appears for the different options such as Normal, Triangle, and Square puck.

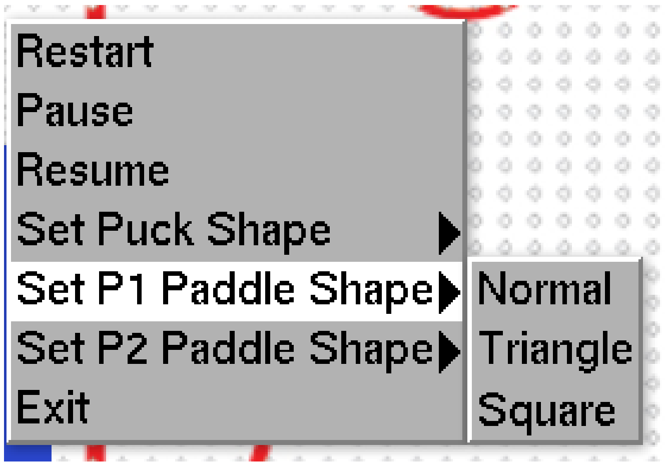

Fig. 5: When set P1 or P2 shape is selected a submenu appears for the different options such as Normal, Triangle, and Square paddles.

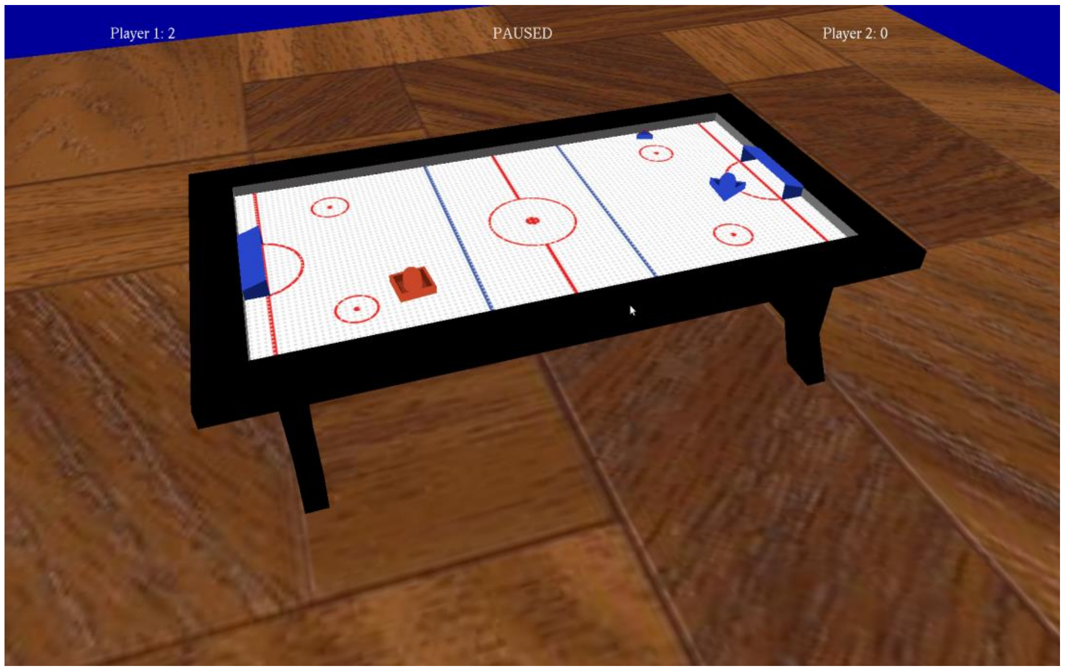

Fig. 6: Different perspectives of the game can be viewed through the movement of the camera, though camera movement is limited. This is also an example of the game being paused, and it displays to the user at the top the game has been paused.

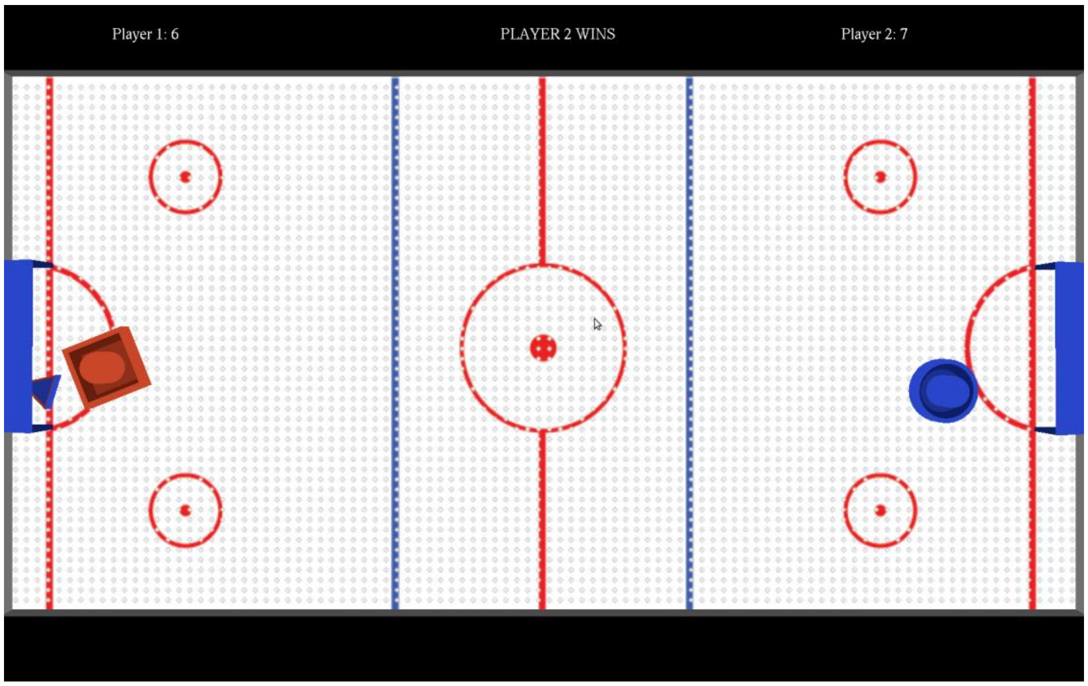

Fig. 7: This is an example of using a square and normal paddle with a triangle puck.

Tech Manual:

Issues:

Table is currently built in pieces as we found our collisions with triangular meshes don't work with concave models. It would map an invisible polygon and treat it as part of the mesh. The way we fixed this was by loading parts of the table and hard coding where they should be placed in the world. All the other types of models worked with collisions by triangular mesh.

There were problems with collision detection because our puck would go through walls. We improved this by turning on continuous collision detection, but a puck can still sometimes go through the walls if moving at very high speeds. The puck resets back on the hockey table if this occurs.

We limited the camera's rotation to keep the mouse controls consistent. If the camera was allowed to rotate fully, then the mouse controls along with the keyboard controls would no longer behave as expected.

Issues that were not resolved:
The puck can force its way under the paddles when both are moving at high velocities. We tried setting linear factor, as well as increasing the height of the puck but both did not work. The issue is not extremely common but can be annoying.

If the paddle is at the location of where the puck will reset after a player scores it will collide with the paddle and cause the paddle to bounce upwards. The bounces will occur over and over until the player moves their paddle off of the puck.

What we would have done differently:
Every object is hard coded into the game, and it would have been easier to have an array of meshes, models, and rigid bodies to handle the objects. Our table was built using different pieces it was especially a hassle to insert into the game.

If we had more time, we would have fine-tuned the player controls. This would consist of making the mouse controls move more smoothly and allowing the keyboard controls to move diagonally. We would have additionally implemented an AI opponent.

The objects are currently large, and due to this the physics are often extreme and do not work as expected. If we had time we would have scaled down our objects.

Electronic copy:
Our Air Hockey can be found at either of the following githubs

https://www.bitbucket.org/chavezmd/cs480chavez

https://www.bitbucket.org/mvancompernolle/cs480vancompernolle