Wii Sports for Dummies

By: Adam Murtagh, Jon Chau, Harris Doan

Lighting, Textures, and Movement

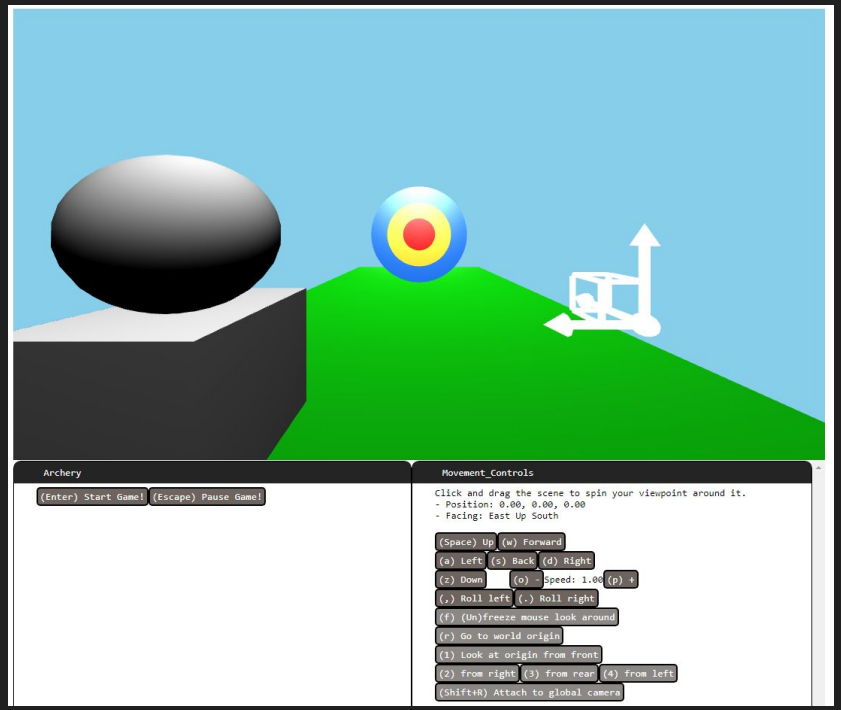
Typical lighting and camera set-up

Generally phong-shaded

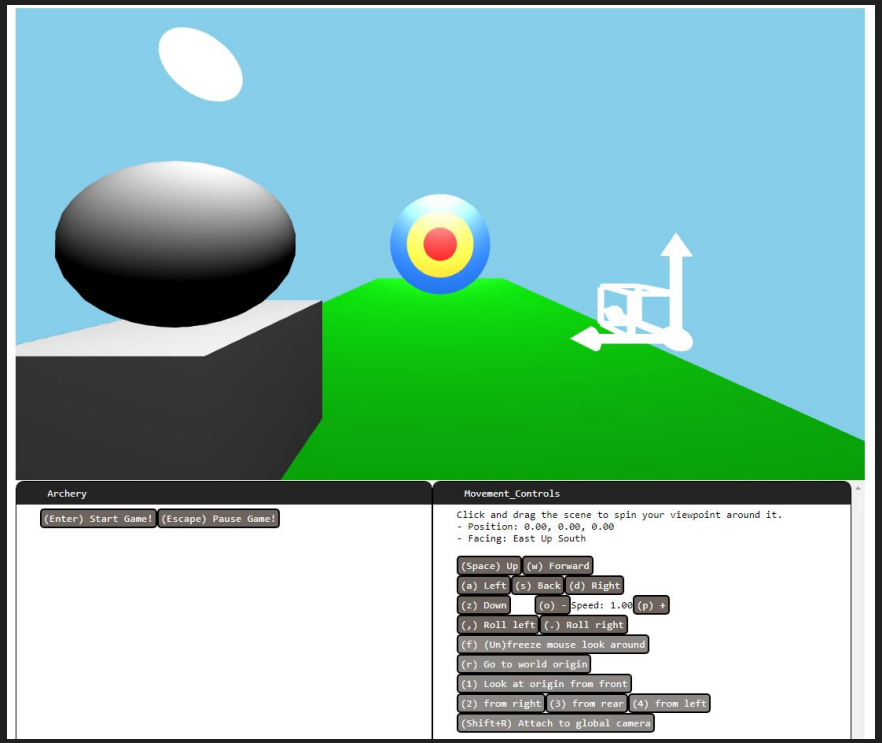
Trees and grassy plains are textured

- Maximum immersion

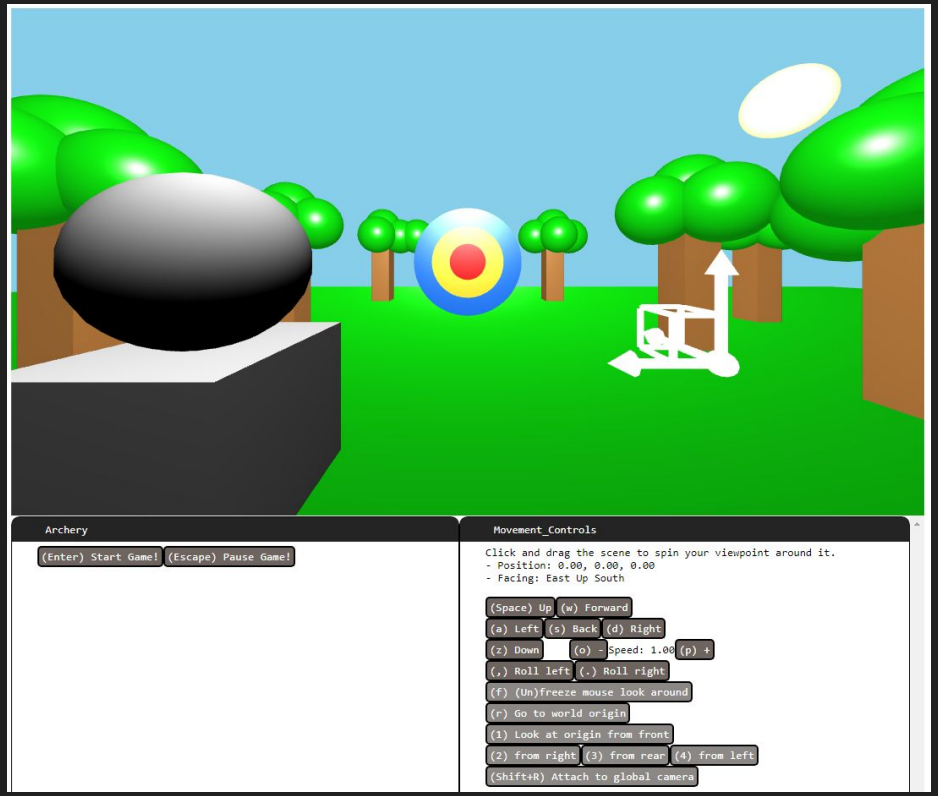
Target and arrow are animated through the use of member variables, time variables, and appropriate physical equations adjusting with each display loop



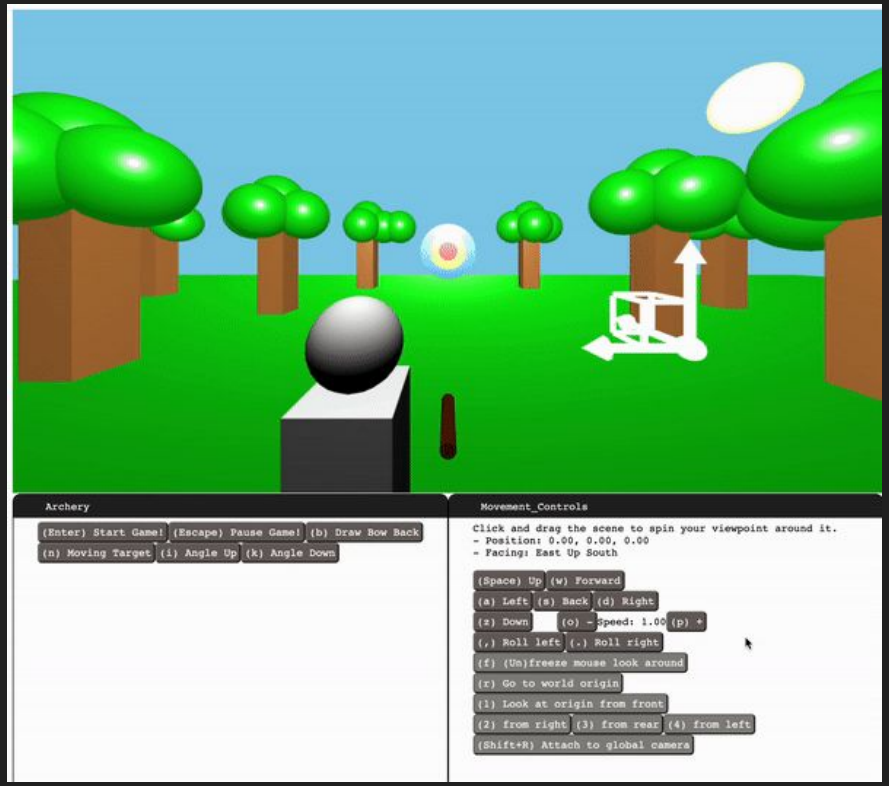
- We initialized the position of the camera and the lighting in our initial stages so that the entire environment would be illuminated.
- We struggled a bit with the initial placement of the “green plane” that would later act as our grass and floor.
- To implement the “blue sky,” we simply drew a blue box that encompassed the entirety of the environment.
- The axis is to help us determine which direction was which.



- We then implemented a sun and would later use that as a source of light to illuminate down on the screen
- However, I felt that since there was already a player model on the left side that the left side of the screen and game felt clutter and therefore unappealing.
- End result had a moved sun to the top right



- After finalizing the sun position, plus adding a yellow ring around the sun to make it look less like a white blob in the sky, the environment was drawn.
- The trees were very easy to reproduce as they were copied and pasted in different positions, this applied to the leaves as well
- The target was also drawn into the environment.



- Now the goal was to add more interactivity into our game.
- The game was quite simple with a static target, static player, and moving arrow.
- So, we decided to implement a moving target.

- In it’s near finalized product, textures were added to the trees and grass to give it a more realistic look then just a flat color.
- The charge bar was added in the upper left for the player to visualize the amount of charge into their bow.

Physics Simulation: Shooting an Arrow

Flight Path

- Initial velocity and angle theta determined by player input
- Calculated through standard sin and cos functions

  y = y + velocity * sin(theta) * time + 0.5 * gravity * time ^ 2

  z = z + velocity * cos(theta) * time

Trajectory

- Tangent of the arrow’s displacements in the y and z axes

Collision Detection

Compares arrow and target locations through the use of member variables

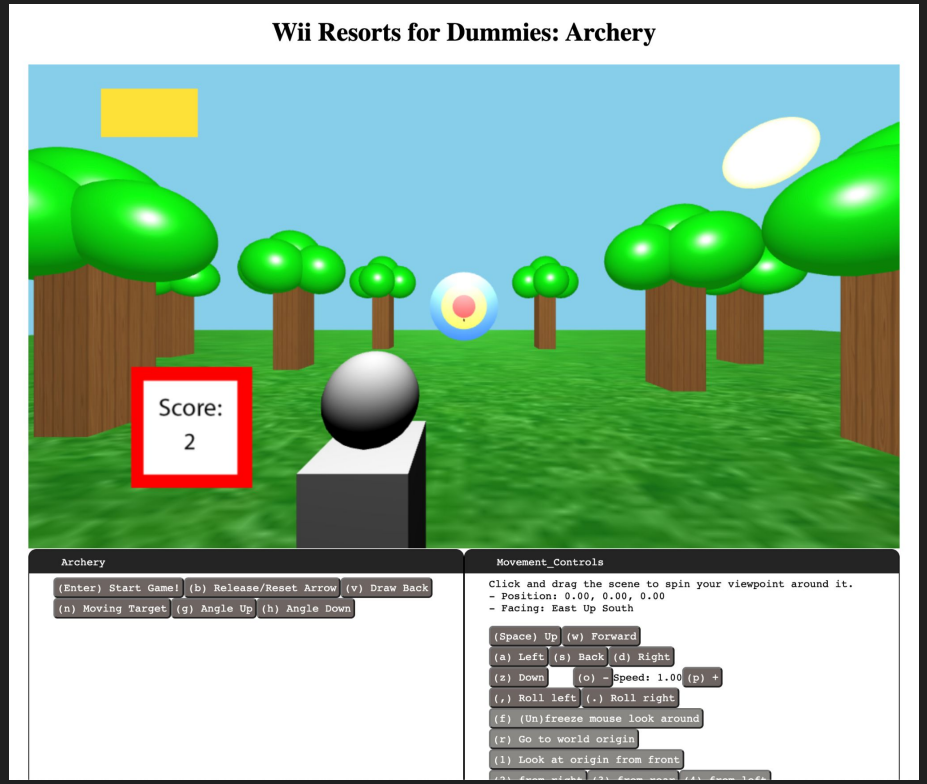
If they are within an appropriate distance, collision is recorded. If the arrow reaches the ground's y coordinate, it is considered a miss

- Arrow "sticks" to target
- Increments score

- With the implementation of the flying arrow and the collision detection, the arrow was able to land on the target.

Scoring

- When an arrow hits the target, the score increments
- Appropriate texture is displayed on a basic square object
- Due to issues with collision, scoring is not dependent on where the arrow hits the target as we had projected



- The final implementation was the working score system

GIF DEMO

Wii Resorts for Dummies: Archery