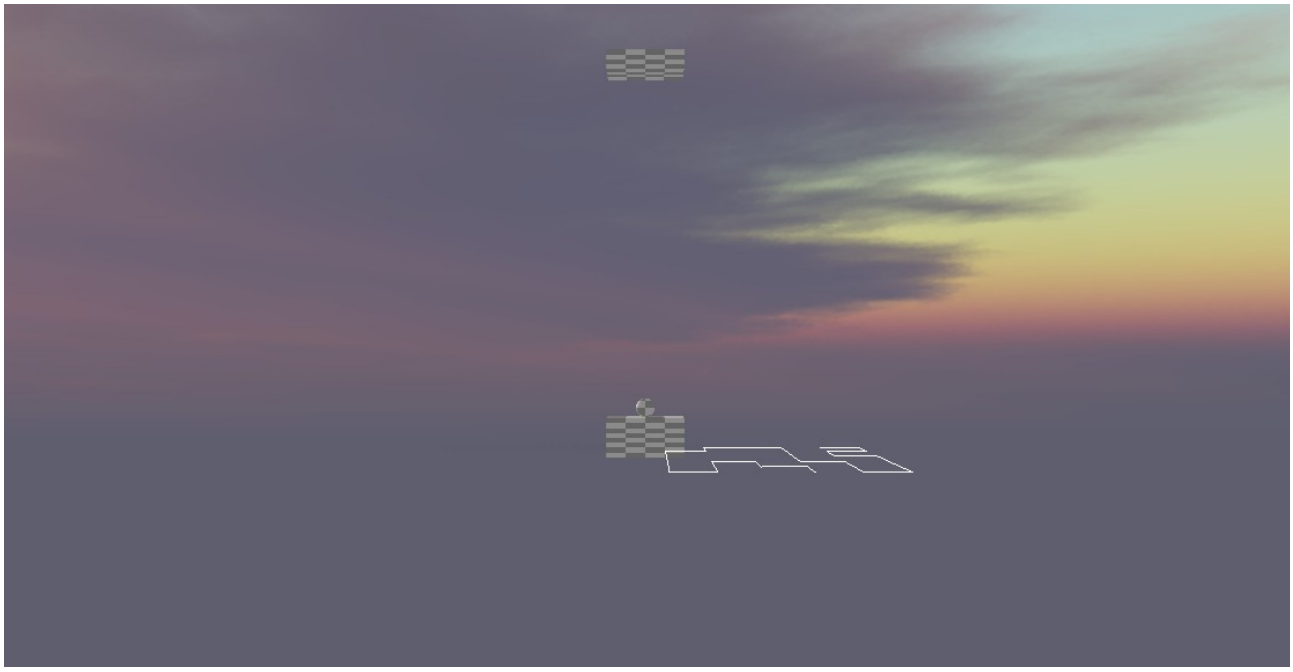
In Part A the player is a sphere that they must bounce to the top platform to proceed through the levels. The player sphere can move and rotate due to impulses from collisions with other objects. Due to the positions of the obstacles and the nature of the gameplay it can be hard to get the player to rotate in practice, but you can force it to happen using the mouse pointer by selecting the player and right clicking towards the edge of the sphere, it will then rotate correctly.



Press the spacebar to control the launcher and try to send the ball into the platform at the top of the stage.

The different collisions shown are:

Sphere / Sphere with the player and obstacle in the 2nd stage.

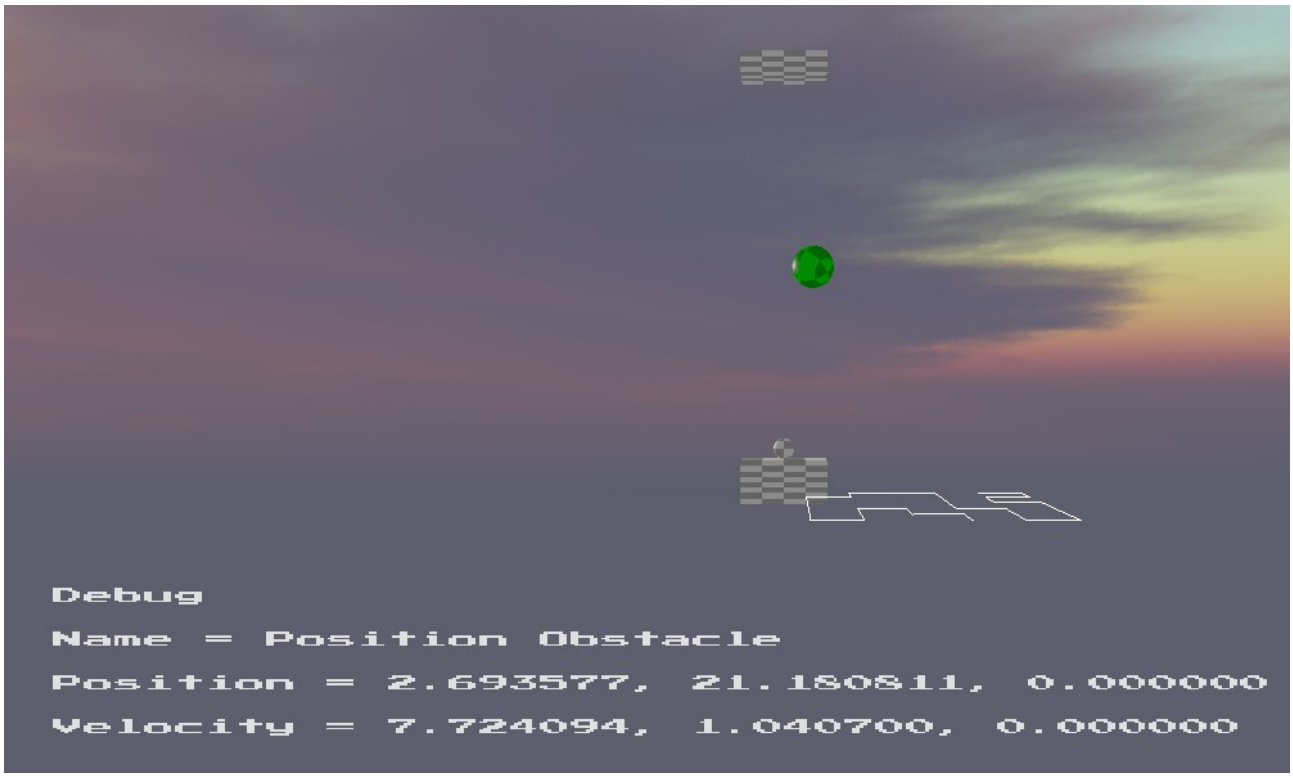
Sphere / AABB with the player vs the launcher and end zone.

AABB / AABB with the launcher resting against the base at the bottom.

Sphere / Plane, there is a killplane under the playable area that ends the game when the player collides with it.

OBB / Sphere with the rotating rectangle obstacle in the 3rd stage.

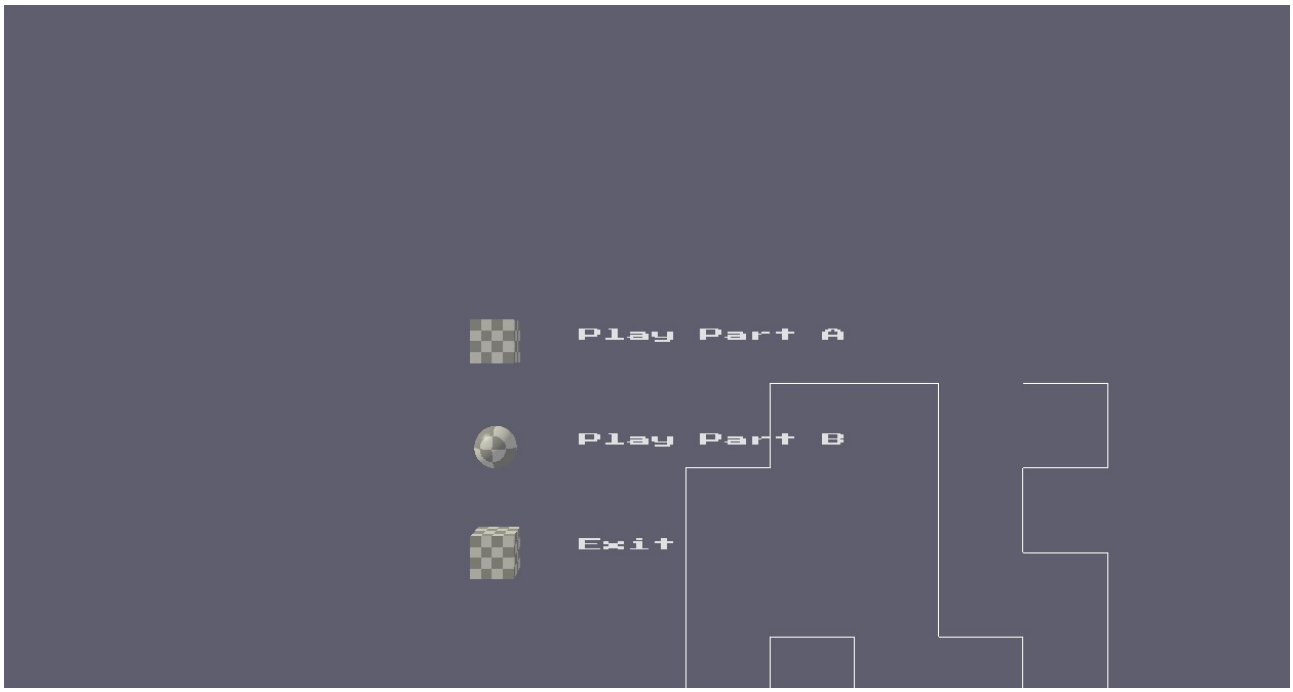
Raycast / World, you can click any object to bring up debug information about it.



In collision resolution both projection and impulses are used, first to seperate the objects and then to give them the correct resulting forces. Also, while friction is not implemented, Elasticity is. The player sphere is more elastic than all the other objects, so it bounces with more force. (The launcher also has low Elasticity).

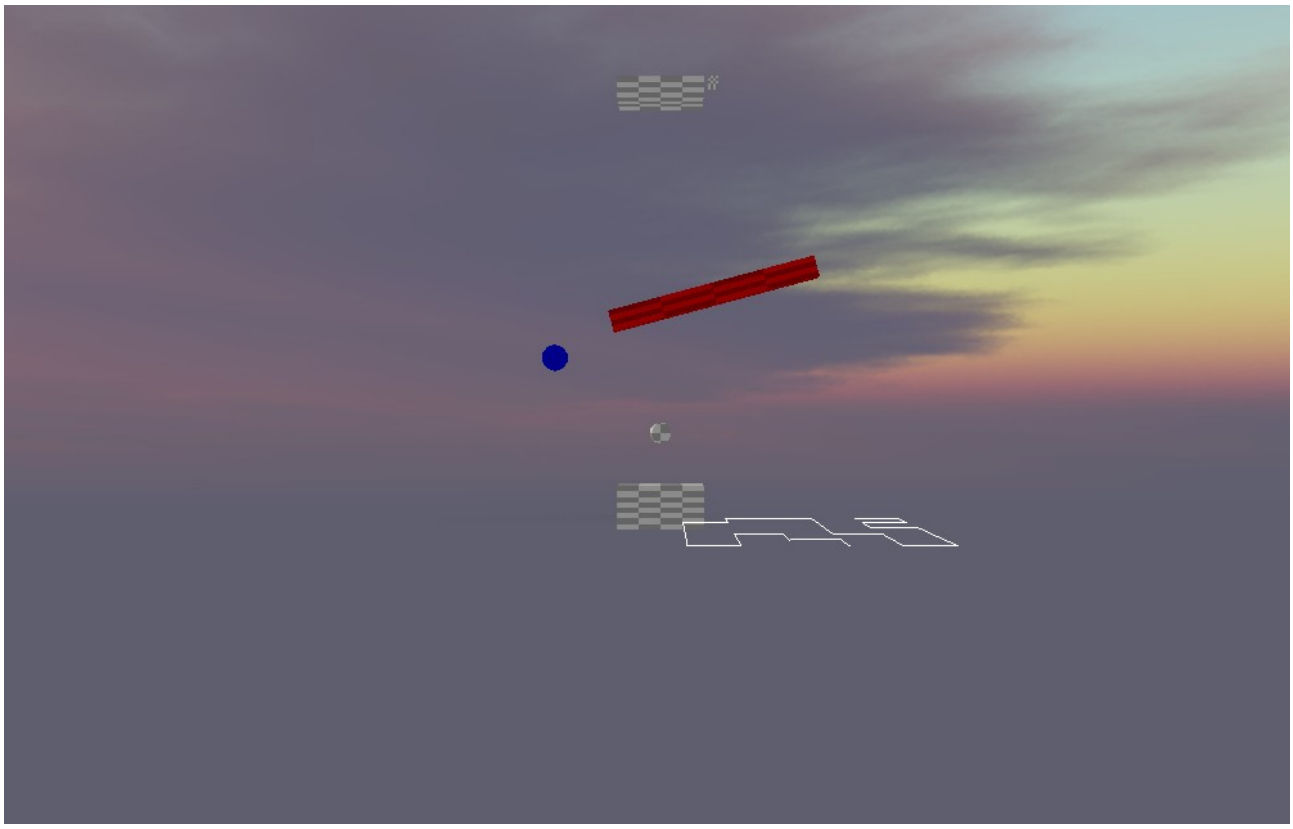
At any time you can press the B key to turn on Broadphase and NarrowPhase spacial acceleration structures. All collision detection should work exactly the same.

There are two different gameplay menus, the main menu and a results screen. You can click one of the shapes next to the writing to navigate through the game.



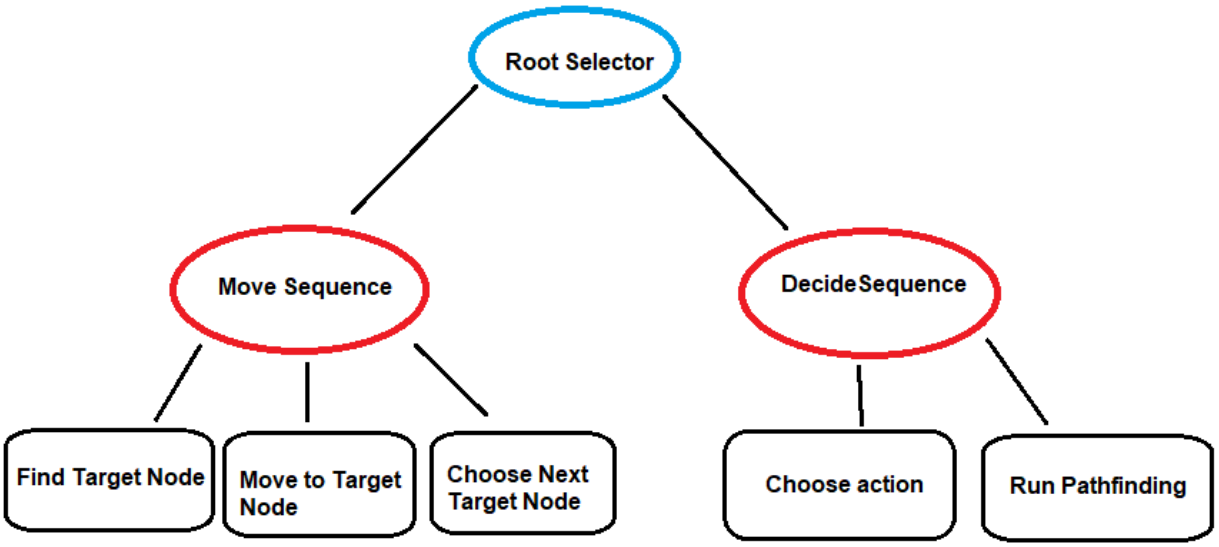
In the game the player can win if they complete all the stages, or lose if they fall from the stage. There is a blue bonus ball in the third stage that gives 50 points when collided with.
When the player either wins or fails the results screen shows their final score and gives them the option to quit or return to the main menu.

The launcher and the swinging ball obstacle in part A both use States and Constraints. A custom position constraint allows for a maximum and minimum distance. And they use states to control which way to move.



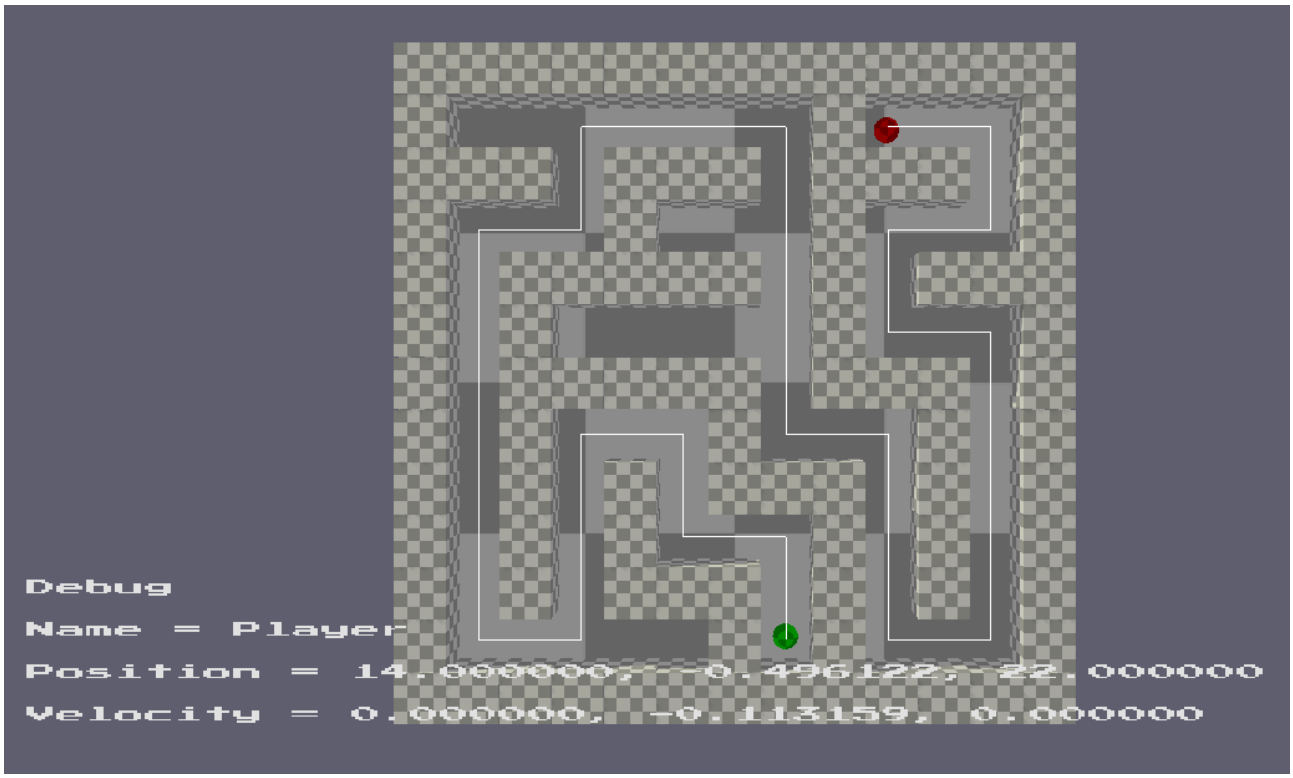
In Part B there is an Enemy AI that uses a behaviour tree and grid based pathfinding. At all times you can see an example of the pathfinding through the maze using debug lines.

Here is a Diagram explaining how the AI is supposed to work.



There is a root selector that first tries to have the ai move to the next node in the path (through the move sequence of actions), if that fails, then it will use the Decide Sequence to run the parthfinding algorithm again. The “Choose action” function will decide what to do (chase player/ wander) then “Run Pathfinding” will get all the nodes to reach that point.

Unfortunately I couldn't get the AI to fully work in time. All the tree sequences and actions are implemented but due to a bug with states the movement code is never run. However all the code is present and the choose action function and the pathfinding algorithm all work correctly (I can see them being run when stepping through code).



Youtube link: https://youtu.be/1agQtYDGJ2A