# CSC8503 – Ori Lazar – b9061712

# Physics Features:

1. Debug Display of physics volumes
2. AABB | Sphere collision
3. OBB | Sphere collision
4. Constrained Objects (OBB Multi-Directional Gate), constraint for height (so don’t sink as not using gravity, since OBBs can’t collide with ABBs….), hinge constraint (orientation of objects must be aligned), Position constraint (so that the objects can’t be pulled away from each other too much).
5. OBB’s as ramps for entering the water.
6. Spring collision for water (penalty method resolution).
7. Impulse + projection resolution with ground.
8. Spring collision (penalty) with walls in the level. Spring stiffness acted upon objects is dependent on the value set by the objects they collide into, so for example the AI colliding into each other (as springs) will result in different stiffness calculation to that of the AI colliding with the walls.
9. Player goose has a fart attack, which applies a force to all the ai units around it

# AI Features:

1. Precomputed A\* pathfinding for both AI types’ paths, I precompute the best paths from each node to every other node, if there is no path, I put -1 in a table, otherwise, I put the index of the next node to travel to, in order to get to a destination node within that table, that way, every frame there is a constant time lookup cost to access the next node, this allows me to include a large number of AI units without any noticeable fps drops. I save this precomputed data into the files (found in assets/data) called “NavigationTableSP.Pathfinding” for the dumb AI units, and “NavigationTableSPKeeper.Pathfinding” for the smart AI units. It should be noted that only a single table must be calculated one time for all AI units using its data.
2. Hierarchical State Machine for Keeper AI (complex AI), has idle, movement and combat states, where the combat state contains a state machine for deciding when to reload / use specific attacks, currently can perform 4 different attacks, 1 of which will result in a player dropping their inventory, the other attacks just applying different forces to the player.
3. Basic AI have 2 states, movement and Idle, where movement will move towards a target if the ai sees a player, or towards its initial (home) position if it stops having a target (if player goes into areas the AI can’t go).
4. New park keeper spawns every 30 seconds and they target the player with the highest score, these park keepers will randomly choose to attack the target player with 1 of their attacks, only one of these will force the player to drop their current inventory items.
5. Debug display of path nodes, and debug display of raycasting lines which are used to target geese (if collided with them).

# Pushdown Automata:

1. Menus run on a pushdown automaton, pushing / popping game states on/off a stack, including the capabilities to extend to leader-board info + dynamic game mode selection once networking is implemented.

# Networking Features:

1. Multiplayer communication capabilities through the server / client system.
2. Data is sent through the network by returning a packet in the update loop, this packet contains a string built summary of that entity’s game time left, the player’s position and the player’s score.
3. Capable of leaderboard integration, as networked end game menus return to the servers and the clients which can then be used to update the leaderboard.

# Misc:

1. To collect items, collide with them and they will be placed in your inventory, the rest of the buttons to press for debug display are shown directly in the game screen at times which you can use them.
2. The world generation is done completely from a file, which was a very handy tool in level design, as I could just escape to the main menu, change my file, and press the start game again, which would then generate a world with the new structure.

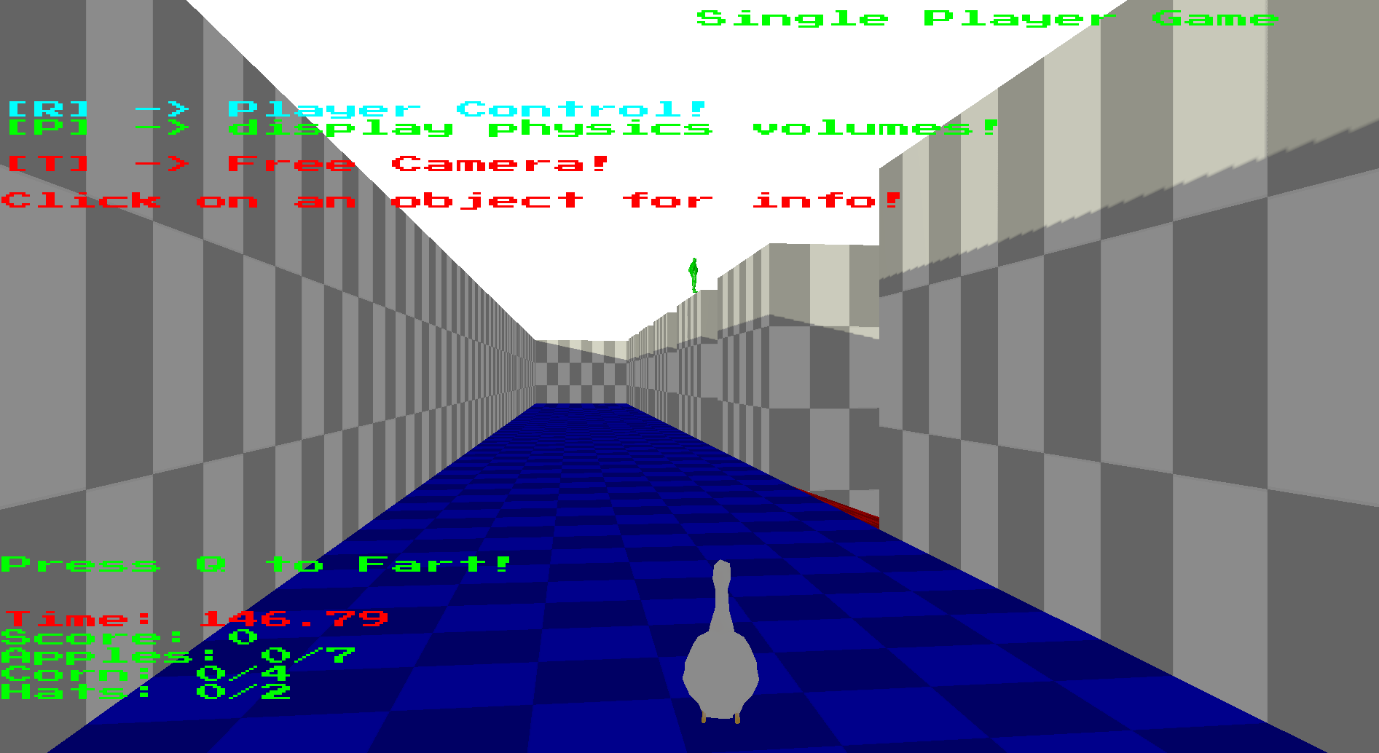
# Keybindings:

**Goose Control:**

1. W – Apply Force Forward (move forward)
2. S – Apply Force Back (move back)
3. A – Apply torque force left (move left)
4. D – Apply force torque right (move right)
5. Q – Let out a massive fart and frighten the ai around you, giving you some more time to escape with their things!
6. R – Toggle on the free movement camera mode!

**Free Camera Mode:**

1. R – Toggle on player control mode
2. P – Display physics volumes mode
3. T – Toggle object selection mode.



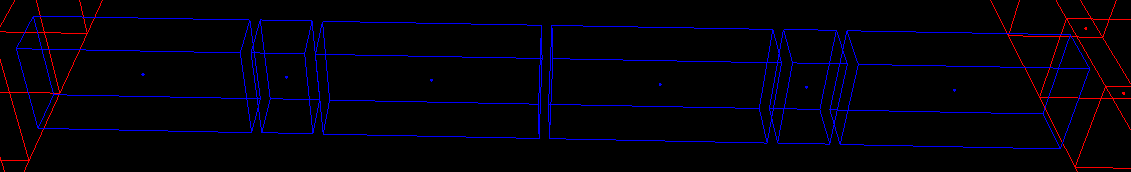
The image above displays the goose control



The screenshot above displays the goose having rotated the constrained multi-gate whilst being attacked by the complex AI which are using a hierarchical state machine to choose which attack to use if they are in the combat state of the parent state machine.



The image above displays the main menu, which is controlled by a pushdown automata.



The image above displays OBB Physics volumes used for the multi-gate constrained objects.



The above image displays the leader board menu.



The image above displays the multi-player select menu, you can choose to either be a server or a client.

The image below displays an example debug display, here I have clicked on the Basic AI object, you can see that it is not a static object, its physics volume is an AABB, its world position is 220, 2.1, 110, its orientation is 0, its FSM is in the idle state, and the description of that state is that no movement occurs. Note that you can also see the raycast lines being performed by the AI at this point, the white and red lines coming out of the ai are its raycast searching for a target.

