Alasdair

Youtube video: https://youtu.be/ZxK5PkyDek4
Youtube video netcoded: *obs only recorded a single window- will try to work this on reupload if able*

Features: Walk around a nav mesh world, in this 'Pikmin' inspired course work. Create your cat swarm by gathering kittens and throw them at dangerous geese wandering around the world.

Menu controls: P: pause, C: start as client, V: start as server, B: start offline
Movement: wasd
Kitten select: hover mouse over location and click with 'left click' mouse button
Kitten throw: 'right click' mouse button when a kitten is near

---

Features as Tick Sheet

Bonus Features-

- ☑ **Kittens move in swarm**: *A simple implementation of boids has been implemented so kittens follow the player as a flock*
- ☑ **Tags, Layers and Trigger Colliders**: *The player can select kittens by hovering their mouse over the kittens. This can be done since collisions can ignore impulse calculations for gameObjects on certain layers. Similarly the cats know when they have collided with the relevant gameObject as a tag system informs them of what they have collided with.*

Player Movement-

- ☑ **Player can move**: *Use wasd to move the player character*
- ☑ **Player can rotate**: *The player rotates to move direction*
- ☑ **Player uses forces/impulses**: *The player moves using addForce methods- as do kittens & all other navMesh agents*
- ☐ **Player uses torque/Impulses for rotation**:

AABB and sphere collision-

- ☑ **Sphere/Sphere:** *Included- same as tutorial*
- ☑ **Sphere/AABB**: *Included- same as tutorial*
- ☑ **AABB/AABB**: *Included- same as tutorial*

Basic Extra collision:

- ☐ **Something vs Plane**:
- ☑ **OBB vs Sphere**: Navmesh world uses OBB- demonstrated with ramps in world
- ☑ **Raycast vs world:** Used in multiple places.
  1. Projecting sphere trigger object to world position from mouse to select kittens
  2. Raycast start point from object into navmesh dimensions (used so navmesh paths can be layered on top of each other)

3. Enemy vision- if there are no collisions between enemy and player then player is considered ‘seen’

Collision Resolution:

- ☑ **Projection method used***: same as seen in tutorial coursework*
- ☑ ***Impulse method used****: same as seen in tutorial coursework*
- ☑ ***Multiple coefficients of restitution****: set and get methods for restitution are included on objects. In this scene spheres have a higher restitution to demonstrate the effect*
- ☐ **Penalty method used***: not included*

Stateful Behaviour:

- ☑ **Simple menu implemented**: Pushdown Automata in Main menu system
- ☑ **Stateful objects in level***: CollectMe objects have state machines that transition between states to grow and shrink the bonus object*
- ☐ **Multiple different obstacle types*****: not included*

Gameplay Effects:

- ☑ P**layer can collect bonuses**: *Player collects points by walking into bonus objects. Each object can be collected once and gives 10 points*
- ☑ **Player can win Game**: *The game is won when all (50) cats are collected and brought back to the green square*
- ☑ **Player can lose Game**: *The game is lost when the player touches an enemy*
- ☑ **Player shown final score**: *The score is displayed on the UI and a menu screen pops up on death with final score*

Advanced AI:

- ☑ **AI Opponent present:** *An AI opponent navigates the world and searches for the player- if a player is spotted it will race towards them*
- ☑ **State based Opponent AI:** *Enemy uses behaviour tree with multiple states*
- ☑ **Behaviour Tree Opponent AI:** *Enemies & Kittens use behaviour tree*
- ☑ **AI using raycasts**: *Mentioned in raycasts section- enemy vision uses raycasts*
- ☐ **AI can collect bonuses***: not included*
- ☑ **AI can avoid Obstacles**: Enemy follows navmesh including ramps without falling off the edge (enemies can also go under existing navmesh paths for tunnel like areas- e.g under the sloped block)
- ☐ **AI can respawn/ teleport if needed***: not included*

Pathfinding:

- ☐ **Grid based Pathfinding present:** *AI Kittens, Swarm and Enemies use navmesh pathfinding, but grid based pathfinding still exists in code base from tutorial*
- ☑ **AI uses pathfinding:** *Enemies path find to waypoints and player*
- ☑ **AI can follow path:** *Enemies follow the path found by the navmesh agent*
- ☑ **Navmesh Pathfinding demonstrated:** *AI Kittens, Swarm and Enemies use navmesh pathfinding*

Constraints:

- [x] **Position constraint demonstrated:** *Bridge from tutorial within project (in the sky)*
- [x] **An obstacle uses constraints:** *Bridge from tutorial within project*
- [ ] **Orientation/ other constraints demonstrated**:

Advanced collision resolution:

- [ ] **Penalty method used:** *Not included*
- [x] **Friction applied during resolution:** *Physics object Friction is applied to contactVelocity during resolution*

Advanced collision detection:

- [x] **Capsule vs Sphere:** *Player cat is a capsule that interacts with sphere collisions*
- [ ] **Capsule vs OBB/AABB:** *Capsule collides with OBB meshes, but logic is the same as Sphere vs OBB no modifications have been made to adjust for capsule start and end*
- [ ] **Capsule vs Capsule:** *Not included*
- [ ] **OBB vs OBB:** *Half working code exists, but other work was prioritized- not included*
- [x] **Spatial Acceleration structures used***: as included in tutorial*

Advanced menu:

- [x] **User can select different game types:** *Start as client, start as server and start offline can each be selected from start which will start the game in a different gamemode*
- [x] **Appropriate handling of menu state:** *States are changed using pushdown automata, pause activates when p is pressed and game starts on starting state.*

Networking:

- [x] **Client can connect to server:** *included as specified*
- [x] **Client can send packets:** *included as specified- client sends acknowledgement packages when receiving a package (these are not used in any way, but they are sent)*
- [x] **Client can receive packets:** *included as specified*
- [x] **Server can send packets:** *included as specified*
- [x] **Server can receive packets:** *included as specified*
- [ ] **Client sends/receives high scores:** *included as specified*
- [ ] **Server sends/receives high scores:** *included as specified*
- [x] **Player Position is sent across network:** *Server player is synced across both clients where the client is in a 'spectate mode'*
- [ ] **Game state is sent via network:** *not included*
- [ ] **Goose (enemy AI) state is sent via network:** *not included*