Professor Please

15-466 P1 Documentation

Anna Etzel (aetzel), Ivan Wang (icw), Jun Huo (jhuo)

Overview

Custom-implemented scripts lie in Assets/Scripts/, while third-party Unity resources are found in Assets/Standard Assets/.

Every character prefab GameObject contains *Movement.cs*, the master class which takes other behavior scripts and scales them appropriately. The resultant vector is applied to ThirdPersonCharacter.Move, a script from Unity standard assets that applies movement to the character.

All behaviors extend *BaseBehavior.cs*, and implement the function
abstract Vector3 ComputeVelocity(),
which is called every frame on Update and returns a new velocity vector based on the behavior. Each behavior also contains a public float scale, which denotes the relative importance of that behavior. Finally, some behaviors are marked with a public bool vital. Vital behaviors (like collision) take priority over others; if their resultant velocity vector is non-zero, all non-vital behaviors are suppressed for that frame.

Testing

The project contains multiple scenes, each titled with the relevant Behavior. Most scenes contain a Professor character, who is player-controlled by the arrow keys. All other characters are Students, who follow specific behaviors and interact with each other. Many students also use the Professor as a target (e.g. ReachGoal or Flee).

Included in the Demo/ folder are also videos, labelled by the relevant Behavior. (You can view .webm files in your browser or using VLC Media Player.)

Behaviors

Wander
Wander algorithm is based on steering behavior proposed by Craig Reynolds. It projects a circle directly in front of the character and pick a displacement from center of the circle. This displacement vector is based on an angle displacement that we increment a random amount each frame. The normalized vector of the vector created from the character's position to the random point on edge of circle is used as the new velocity. Character's movement is kept generally forward, while adding some randomness in turning. This creates smooth, random turns.

ReachGoal
The ReachGoal behavior is implemented by checking the target's position vector and subtracting the current transform's position vector. This is appropriately scaled based on the distance (when the character approaches the stopping distance (radius), its velocity is scaled down accordingly). When the character is past the stopping distance, it ceases to move.

If the character is far enough away and has "lookAhead" enabled, then it will attempt to intercept the target by taking the target's position vector + velocity vector as the actual target.

FlockingWander
For each follower, a position in an arc around the previous generated position at some random distance between two variables is chosen. This leads to reasonable clustering, but maintains (usually) a separation between the followers. The characters move towards their position in the formation relative to the leader. The leader simply uses Wander behavior, so the entire group simulates a random flocking wander.

Flee
The Flee behavior is the opposite of ReachGoal, taking the vector away from the target position. Similarly, velocity is scaled according to how close they are to the stopping distance. When the character is past the distance, it switches to an "idle behavior," which defaults to wander.

FleeFromGroup
Similar to Flee, this behavior keeps track of an array of targets. For each target, it computes the velocity away from and sums them all together. If the target is too far away, it is ignored. If all targets are far, it switches to the "idle behavior."

Formations
For formation behaviors, the Student{Shape}FormationLeader has an attached script {Shape}Formation.cs which assigns positions relative to the leader for each follower. The

followers are listed in an array in this script. Each follower has the FormationFollow BaseBehavior attached, making them go towards their assigned position.

Collisions

We cast a constant number of rays in a cone shape (approximately 40 degrees) from slightly behind each character. If the ray collides with any GameObject tagged “Wall”, the character turns an arbitrary direction to avoid the object. Because this behavior is marked vital (see Overview section), collision avoidance takes priority over other behaviors.