Inteligência Artificial para Jogos

2020/2021

Laboratory Guide

Lab 2 – Collision Avoidance and Movement Arbitration

1. Explore the provided scene

a. Open the downloaded project in Unity and explore the scene *Priority and Blending* . This scene will allow us to explore collision avoidance and to understand the difference between combining movement using a Blending or a Priority mechanism.

b. Analyze the code that implements the movement combination/arbitration mechanisms.

c. Add the DynamicArrive behaviour you created in the previous Lab and integrate it in both the Blending and Priority Mechanism for the MainCharacterController. You should now be able to use the DynamicPatrol Movement.

2. Avoiding collisions with static objects

a. Implement the movement DynamicAvoidObstacle in order to avoid collisions with static objects. Start by using a detection mechanism based on a single ray. You can use Unity's collision detection mechanism. If you execute the method gameObject.GetComponent<Collider>()[1] with a unity's GameObject you will get a collider that can be used to verify if a specified ray collides with the object.

b. Add the avoid obstacle behaviour to both the blending and the priority mechanisms of the MainCharacterController. Try out different weights for the blending behaviours. Observe the difference in avoiding obstacles when using priorities and blending. Which one will get the best results? Why?

c. Implement a new version of avoiding obstacles, but this time using a central ray with short whiskers. Compare both versions. Which one obtains the best results? Why?

3. Avoiding collisions with moving objects

a. Do you think that the previous collision avoidance algorithm can be used to avoid collisions with other moving characters? Try to explain why.

b. Implement the movement DynamicAvoidCharacter in order to avoid collisions with moving characters. Use the algorithm described in the theoretical classes.

c. In order to properly test the behaviour, you will need to add more characters to the scene. Adjust the collision avoidance parameters so that the characters will only change direction when needed. Test the resulting behavior to see if everything is working as intended.

d. Open Unity profiler, create 50 characters, and determine which method/methods are consuming most time each frame.

[1] Take into consideration that the GetComponent method may not be very efficient, so we do not recommend its use in all frames.