GD2P01 – Artificial Intelligence for Games

Mini-Project (30%)

Component code and name	GD2P01 Artificial Intelligence for Games
Assignment name	Assessment 2 – Mini Project
Weighting	30%
Submission deadline	Week 11
Week issued	Week 7

Brief

This assessment will demonstrate that you understand the theory and implementation of Steering behaviours that can be used by an agent to navigate the environment. You will use the fundamental building blocks of software engineering that you have learned before, together with the steering behaviour algorithms to create an environment that utilises these different behaviours to move an agent/group of agents around.

Please refer to the Task Instructions (below) for further details on how to complete this task.

Instructions/Requirements

- Using **C++ and any needed libraries**, create an environment (scene) in which **six** of the following Movement techniques are demonstrated:

1. **Simple behaviours for individuals and pairs (pick any four of the following):**
    1. Seek and/or Flee
    2. Pursue and/or Evade
    3. Wander
    4. Arrival
    5. Obstacle Avoidance
    6. Containment
    7. Wall Following
    8. Path Following

2. **Combined behaviours and groups (one)**
    - Implement the Flocking algorithm (combining separation, alignment, and cohesion)

3. **Combined behaviours and groups (pick only one of the following):**
    1. Crowd Path Following
    2. Leader Following
    3. Queuing

- A Readme file, that clearly specifies which of these behaviours you have implemented within your application.

Application Design Requirements:

- All algorithms need to fulfil the following tests and rules.

Behaviour	Tests
Seek/Flee	• User should be able to control the target movement. • Source must display Seek/Flee behaviour towards the target • Must follow Border Rule
Pursuit/Evade	• User should be able to control the target movement. • Source must display Pursuit/Evade behaviour towards the target. • Must follow Border Rule
Wander	• Object to display the wander behaviour. • When clicked on the screen, a new object gets spawned at that point and displays wander behaviour. • Must follow border Rule
Arrival	• User should be able to control the target movement. • Source must display Arrival behaviour towards the target • Must follow Border Rule
Containment **Wall Following** **Path Following** **Obstacle Avoidance**	• An object to display intended behavior • When clicked on the screen a new object gets spawned at that point and displays respective behaviour. • Must follow Border Rule if applicable.
Flocking	• A flock of minimum 50 individuals. • When clicked on the screen, a new object gets spawned at that point, joins the flock, and displays flock behaviour. • Must follow Border Rule
Leader Following	• User should be able to control the Leader movement. • Minimum 15 followers. • Must follow Border Rule
Crowd Path Following	• A crowd of minimum 50 individuals. • When clicked on the screen, a new object gets spawned at that point, joins the crowd, and displays crowd Path following behaviour. • Must follow Border Rule if applicable

Queueing	• Minimum 50 individuals. • Exit point should be at the end of the screen. • No more than 5 individuals must go through the exit point • When clicked on the screen, a new object gets spawned at that point and joins at the back of the queue. • Must follow Border Rule.

*Border Rule -> If an object moves out of the screen boundary, it must wrap around to the opposite side.

The scene should include (but is not limited to) the following functionality.

- The user can add more agents within the scene.
- The user can choose which algorithm the agent(s) will follow.
- The environment has some obstacles that the agent needs to avoid.
- The movements of the agent(s) are smooth, and they do not get stuck around corners or on the walls.
- If your object moves out of the screen boundary it needs to wrap around from the opposite side (Border Rule).

Guidelines

Build Quality:

The source code is required to display the following qualities:

- Free of:
  - Build errors and warnings
  - All intermediate files
- Commenting, Naming, Structure & Documentation:
  - Code formatting is consistent, with good use of whitespace, tabbing and alignment.
  - Consistent and clear naming conventions are used.
  - Where necessary, **comments** should be used to clarify the purpose and use of data and functions. **Comments in the context of this assessment will also demonstrate understanding and provide evidence of academic integrity.**
  - Any necessary documentation should be included as a separate Readme.txt file.
- An electronic copy of the source code is required.
  - Name the source code folder as: Source – Student Name

Interface Features:

The executable is required to provide an intuitive interface with the following features:

- Provide clear instructions.
- Controls are clearly identifiable and intuitive while playing.
- Design and layout of the UI make effective use of screen space.

Runtime Quality:

The source code is required to display the following qualities:

- Free of:
  - Bugs.
  - Crashes.

Submission Guidelines

Zip files containing the following.

- **Build Zip:** An executable must be zipped and included with the submission.
- Readme.txt with details about the techniques used and any other dependencies.
- **Source Code Zip:** All relevant source code files and project files must bezipped and included with the submission.

An electronic copy of the source folder including only source code is required as described in the buildquality section.

Submit the exe zip as described in the executable build section.

Submit the appropriately named zip file to 'blackboard' with the following structure:

Naming & file structure for the zip file.

- Assessment2_StudentName.zip
  - Build.zip
  - SourceCode.zip
  - Readme.txt

Assessment Criteria

- Implement six steering behaviours (70%)
- Non-Core features (20%)
- Software organization (5%)
- Naming conventions and coding Standards (5%)

Submission Policy

Please refer to the Submission Policy on Blackboard for further details.