BOID SIMULATION

UNIVERSITY OF BUCHAREST

COMPUTER GRAPHICS 2D PROJECT

PĂCURARIU RĂZVAN MIHAI LUPARU IOAN TEODOR

2025-2026

CONTENTS

CONCEPT

The scope of this project was to implement a boid simulation, alongside some other features, in an easily extensible way.

USED TRANSFORMATIONS

This project uses all the standard glm transformations a 2D OpenGL project might need. The ortho transformation is used to properly map the entities' coordinates onto screen space, alongside the translate, rotate and scale transformations, which are used both for the camera and the entities to change their position, orientation and zoom/size in the 2D space.

DESCRIPTION AND ORIGINALITY

The project includes, alongside the already mentioned standard boid simulation, nests. On initialization, each boid is assigned a random nest to which is attracted to, as in its velocity vector is guided towards the center of the nest, alongside the usual components that are added to it by a boid simulation. When a boid gets too close to a nest, it will try to orbit it (and on certain simulation parameters, it will also achieve this stably).

At a constant interval, a random boid is chosen for their nest and the nests of all its nearby neighbors to be reassigned to the same random one. This encourages more dynamic movement that displays the swarming effect of the boids more clearly.

There are clouds that pan from left to right and loop back around to accentuate the feeling of a top-down perspective of some birds. These clouds move at different speeds, based on a randomly chosen height value, and also have random a random scale and rotation, giving the illusion of a more diverse selection of cloud shapes.

As for controls, there is the ability to zoom in or out using the scroll wheel, and also to track either a random boid by pressing T or a random nest by pressing Y (pressing T/Y again will return the camera back to the standard view). There are also three sliders at the bottom of the screen, which control three of the main parameters of the boid simulation, encouraging the user to interact with the application for longer. These sliders are controlled using the `[`, `]` keys, `;`. `'` keys and `,`, `.` keys respectively. The shaders are rendered with a custom (pair of) shader(s), giving them a border, and also allowing the slider bar length to be changed using a uniform.

On the backend side of things, a mini engine was implemented to allow for easier work with OpenGL, there being a Renderer which can render and update a Scene which stores a list of Entities. Each Entity has a Mesh, which stores the VAO, VBO, and EBO for the entity, and it can also choose which shader to use. There is also a TextureManager that manages textures and an InputManager that allows for easy bindings between an input method and a lambda. Other classes present in the engine are the Camera(representing the camera in a scene, stores position, rotation, zoom, other helper functions and members (such as for tracking)), Texture(interface over the usual way to create a texture in OpenGL, easy to interact with), Shader(class abstraction over loadShader.cpp), App(class representing the application itself, handles properly initializing the Window and Renderer and so on, also the actual scene content and input bindings are specified in it), Window(handles creating the GLUT window and also stores some info about it).

This engine was designed to be extensible, as can be seen by the boid simulation being represented by a BoidScene, class derived from Scene, and the Boid, Nest, Cloud and Slider classes, derived from the Entity class. For the derived entities in the boid scene in particular, they use shared resources such that all boids use the same Mesh and Texture data, similarly for nests and clouds, to save on memory usage. The clouds use 4 different textures which are chosen at random upon initialization.

All the textures present in this application were hand made by the team members and were not taken from an asset site.

IMAGES

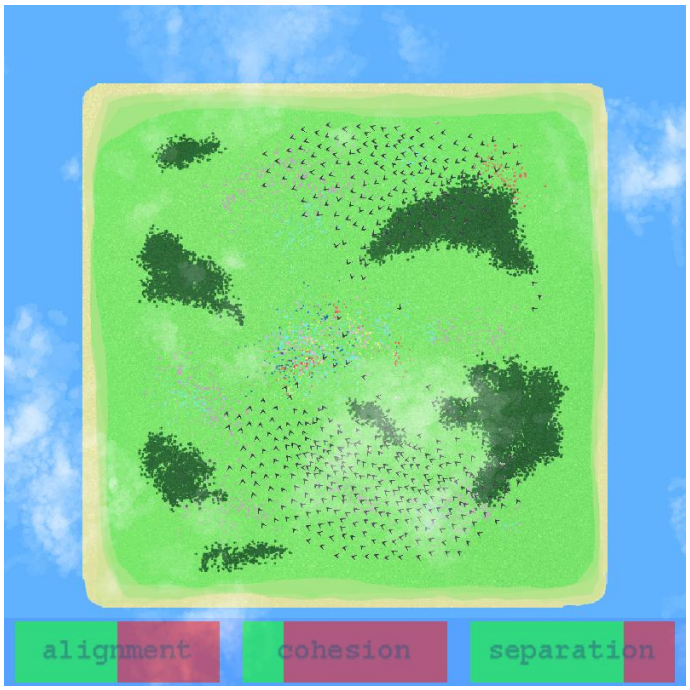

Figure 1: The main view of the application. In it can be seen the boids moving as expected, nests and clouds, alongside the 3 sliders at the bottom.

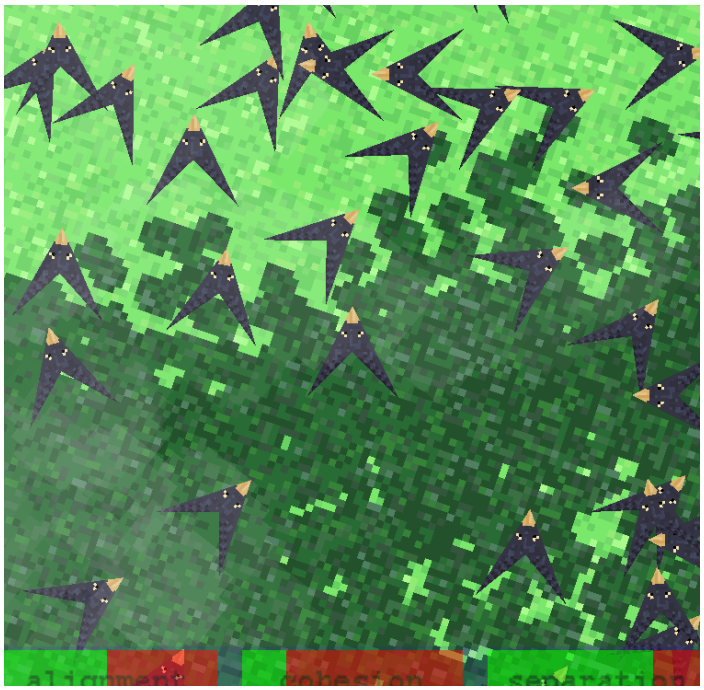

Figure 2: Cropped and zoomed in view of a boid being tracked.

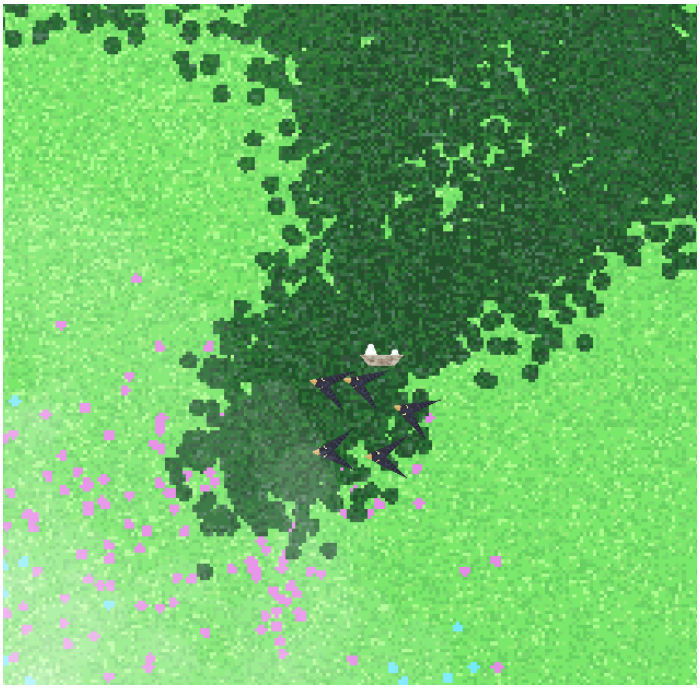

Figure 3: Cropped and zoomed in view of a nest. Boids can be seen circling it.

INDIVIDUAL CONTRIBUTIONS

Păcurariu Răzvan Mihai implemented the engine, the initial boid simulation and sliders.

Luparu Ioan Teodor implemented the nests, clouds, and also helped with coming up with the ideas for this project.

Both worked a roughly equal amount on the textures.

GITHUB

The repo for this project can be found at https://github.com/razvanpacku/ComputerGraphics-Project-2D.

REFERENCES

- This site was used as a reference for implementing the boids algorithm.
- The usual sources of help for fixing problems or figuring out how to do something (StackOverflow, LLMs, VS's autocomplete and copilot).

APPENDIX: SOURCE CODE