CSC413 Final Project: Boids Ecosystem

Eric AhSue
CSC413 Fall semester 2024

github.com/Jasuv/CSC-413-Final_project

Introduction

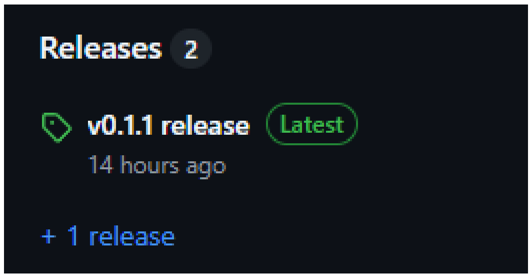
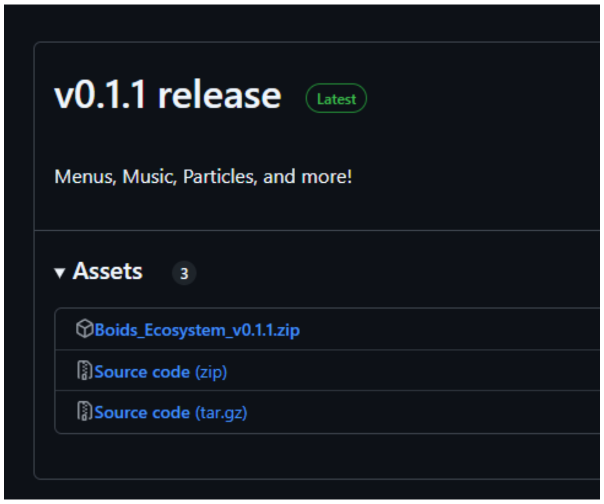
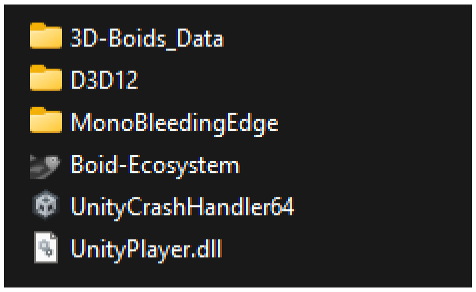
For this term project, I began by researching Boids, a natural simulation developed by Craig Reynolds to model flocking and schooling behaviors in birds and fish. The concept and implementation were pretty straightforward: each boid would move towards the center of mass of the flock, avoiding collisions with others, and align themselves in the direction of its neighbors. These simple rules create a beautiful demonstration of emergent behaviour. But the challenge came from optimizing the algorithm. In my initial version, each boid would consider the position of every other boid, which skyrocketed the number of calculations for every new boid added. My first attempt at optimization, I used sphere casts so each Boid would only "see" other Boids in that radius. But even then, the maximum number of boids I could spawn was stuck around 500. So I explored alternative solutions and discovered Unity compute shaders. Shaders, written in languages like GLSL or HLSL, allow the GPU to perform complex tasks such as simulating light interactions by processing thousands of simple calculations in parallel. This was the perfect solution to my optimization problem. After adjusting my code to utilize a compute shader, the performance significantly improved, enabling me to render 20x more boids while still maintaining a stable framerate.

With this breakthrough, I realized I had the opportunity to create a living ecosystem using boids, which led to the development of my Boid Ecosystem project. The game functions as a sandbox simulation where the goal is to cultivate the environment to support fish-like creatures by growing their food source and witnessing their evolution. I also wanted to emphasize creating beautiful and diverse sceneries, though this aspect is severely limited by the lack of content. Currently, the game features only one food source (kelp), a single boid body type, and basic terrain (no tunnels/caves), but my vision for this project is to create an ocean life simulation game, where emergent ecosystems drive the experience.

Development environment

Development primarily happened on the new *Unity game engine v6.0* where I coded my animations and shaders, but the models were made in the *Blender modeling software*.

How to download

 Releases 2 v0.1.1 release Latest 14 hours ago + 1 release	Head to the repository linked above, and go to the *Releases* page on the right.
 v0.1.1 release Latest Assets 3 Boids_Ecosystem_v0.1.1.zip Source code (zip) Source code (tar.gz) Menus, Music, Particles, and more!	Here, you can download the latest version of the game by downloading the zip folder
 3D-Boids_Data D3D12 MonoBleedingEdge Boid-Ecosystem UnityCrashHandler64 UnityPlayer.dll	Unzip the folder and launch the application. You're all set!

How do you play

In the game, there’s an interactive tutorial that teaches you the movement, game UI, mechanics, and how to play. Here are the excerpts from that tutorial.

Welcome to Boids Ecosystem! The goal of the game is to create a suitable habitat for the fish by balancing their population and food source. As you watch them evolve, notice how new species may appear and their behavior change!

Movement:
Use [W] [A] [S] [D] to move around and [Q] [E/space] to go up and down.
Drag using [Right Mouse Button] to rotate your camera.

Speed Control:
At the bottom right you will see 4 buttons that adjust the game speed by:
0x, 1x, 2x, and 3x

Getting started:
The first step of the game is to start planting the food source. Use [Left Mouse Button] to start planting seeds on the sea bed. Planting seeds costs points (2).

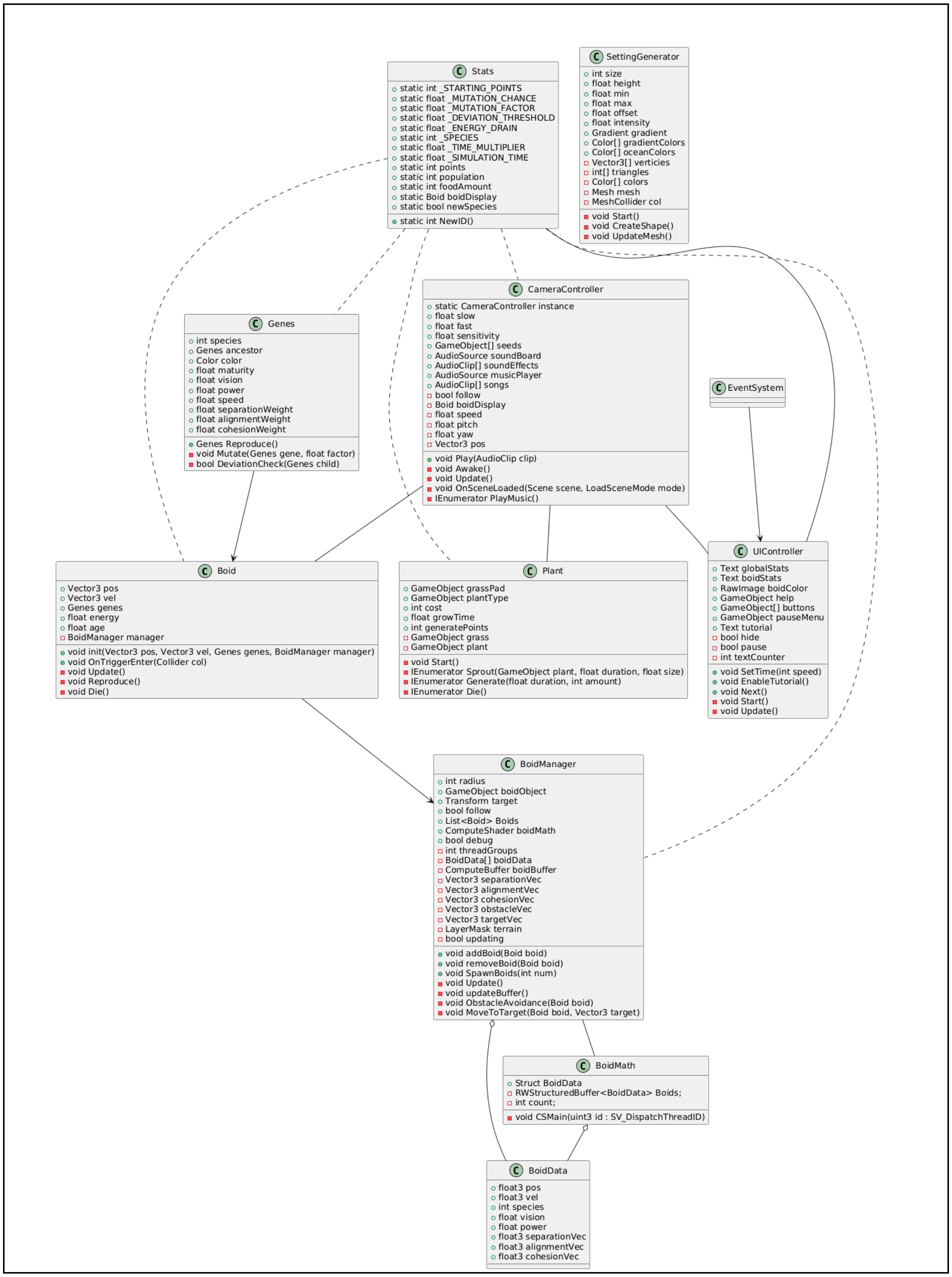
Your seeds will begin to grow into beautiful seaweed! This seaweed will serve as the primary food source for the fish. While growing, they will start generating more points.

Fill the Tank:
Once you have at least 30 points, press the Fish button on the top left to spawn some fish! These fish will evolve over generations and take on new traits.

You can click any fish you want to view their stats. You can also press [F] while selecting a fish to follow them.

Fish Behavior:
Each fish has an energy and age stat that determines their survivability. Most fish typically die at 100, but require energy to make it that far. Make sure there is enough food for everyone!

Once a fish hits age 30, they will have a chance to (asexually) reproduce! The children will have a chance to alter their genome. If a new strand is significantly different from their ancestors, they are considered a new species!


Stats
static int _STARTING_POINTS
static float _MUTATION_CHANCE
static float _MUTATION_FACTOR
static float _DEVIATION_THRESHOLD
static float _ENERGY_DRAIN
static int _SPECIES
static float _TIME_MULTIPLIER
static float _SIMULATION_TIME
static int points
static int population
static int foodAmount
static Boid boidDisplay
static bool newSpecies
static int NewID()
SettingGenerator
int size
float height
float min
float max
float offset
float intensity
Gradient gradient
Color[] gradientColors
Color[] oceanColors
Vector3[] verticies
int[] triangles
Color[] colors
Mesh mesh
MeshCollider col
void Start()
void CreateShape()
void UpdateMesh()
CameraController
static CameraController instance
float slow
float fast
float sensitivity
GameObject[] seeds
AudioSource soundBoard
AudioClip[] soundEffects
AudioSource musicPlayer
AudioClip[] songs
bool follow
Boid boidDisplay
float speed
float pitch
float yaw
Vector3 pos
void Play(AudioClip clip)
void Awake()
void Update()
void OnSceneLoaded(Scene scene, LoadSceneMode mode)
IEnumerator PlayMusic()
Genes
int species
Genes ancestor
Color color
float maturity
float vision
float power
float speed
float separationWeight
float alignmentWeight
float cohesionWeight
Genes Reproduce()
void Mutate(Genes gene, float factor)
bool DeviationCheck(Genes child)
EventSystem
Boid
Vector3 pos
Vector3 vel
Genes genes
float energy
float age
BoidManager manager
void init(Vector3 pos, Vector3 vel, Genes genes, BoidManager manager)
void OnTriggerEnter(Collider col)
void Update()
void Reproduce()
void Die()
Plant
GameObject grassPad
GameObject plantType
int cost
float growTime
int generatePoints
GameObject grass
GameObject plant
void Start()
IEnumerator Sprout(GameObject plant, float duration, float size)
IEnumerator Generate(float duration, int amount)
IEnumerator Die()
UIController
Text globalStats
Text boidStats
RawImage boidColor
GameObject help
GameObject[] buttons
GameObject pauseMenu
Text tutorial
bool hide
bool pause
int textCounter
void SetTime(int speed)
void EnableTutorial()
void Next()
void Start()
void Update()
BoidManager
int radius
GameObject boidObject
Transform target
bool follow
List<Boid> Boids
ComputeShader boidMath
bool debug
int threadGroups
BoidData[] boidData
ComputeBuffer boidBuffer
Vector3 separationVec
Vector3 alignmentVec
Vector3 cohesionVec
Vector3 obstacleVec
Vector3 targetVec
LayerMask terrain
bool updating
void addBoid(Boid boid)
void removeBoid(Boid boid)
void SpawnBoids(int num)
void Update()
void updateBuffer()
void ObstacleAvoidance(Boid boid)
void MoveToTarget(Boid boid, Vector3 target)
BoidMath
Struct BoidData
RWStructuredBuffer<BoidData> Boids;
int count;
void CSMain(uint3 id : SV_DispatchThreadID)
BoidData
float3 pos
float3 vel
int species
float vision
float power
float3 separationVec
float3 alignmentVec
float3 cohesionVec

Class Descriptions

Boid Algorithm

- Boid
    - Keeps track of the Boid’s position, velocity, and genes
    - Main behaviours: gain/deplete energy, reproduce, die
- Genes
    - Keeps track of traits/species/ancestors
    - Slight chance to mutate at birth
    - Determines if gene is significantly different from ancestor gene
- BoidManager
    - Manages all Boids and their velocity
        - Algorithm force + Object avoidance force + Follow target force (didn’t use)
    - Can spawn boids
- BoidData
    - Struct to transfer Boid info to compute shader
        - e.g vector3 -> float 3
    - Keeps track of Cohesion, Separation, and Alignment forces
- BoidMath
    - Compute shader to do the Boid Algorithm
        - All boids will search for nearby boids and calculate their respective Cohesion, Separation, Alignment forces

Controllers

- Stats
    - Tracks all stats in static variables
        - e.g population, species count, food count, mutation chance, etc.
- CameraController
    - Player functionalities
        - Movement
        - Boid interaction
        - Planting
        - Other commands (restarting, hiding UI, etc)
- UIController | EventSystem
    - Handles all UI functionalities (EventSystem is Unity’s built-in UI interaction handler)
    - General/Boid stats, Speed controls, Interactive tutorial, Spawn Boids button

Other classes

- Plant
    - Handles plant growth, point generation, and death animation
- SettingsGenerator
    - Randomly generates terrain
    - Chooses random ocean color

Assumptions Made

The biggest assumption I made was expecting players to be patient. The game starts slowly as you work on cultivating the kelp forest, and progress is measured over generations as the boids evolve and develop new traits. The game's appeal lies in this gradual progression, where the journey itself is the primary source of entertainment. I hope players find the journey fun. I also forgot about one minor detail: there are no restrictions on how far the player can move the camera, meaning it's possible to go out of bounds with no easy way to return without quitting the main menu and restarting.

Self-reflection on Development process

Development proved to be challenging, as every time I introduced a new feature, I inadvertently broke my body logic—whether it was improperly balancing force vectors, failing to buffer boid data, or encountering other issues. Balancing resources and boid energy was also tricky, as I needed to ensure the game didn’t feel either too boring or too intense. I had many ideas for additional content to expand the gameplay and make the intended slow pace feel less limited in interactions, but I ran out of time. I plan to address these challenges and refine the game in the future.

Also my playtesters noted a few bugs that slipped past my own testing phase that unfortunately didn't get fixed before submission:

- Changing scenes (e.g game > main menu > game) doesn’t reset the general stats
- Clicking anywhere on the screen while following a boid completely breaks the Player’s controls (except for [Esc] to pause and quit to main menu)
- Sometimes 3x is too fast for the Boids and they clip out of bounds, dying to the killbox I implemented for that exact reason
- Controls menu accidentally says [W] [D] for forwards/backwards, which is supposed to be [W] [S]

Project Conclusion

This project has been an absolute blast to work on. I've come to appreciate the simplicity yet complexity of the emergent behavior from the three basic rules of cohesion, separation, and alignment. One of my most memorable moments was discovering how to use compute shaders to optimize the boid algorithm; it was so mesmerizing watching tens of thousands of Boids fly across my screen in beautiful flocking patterns. Another major accomplishment was writing the shader code for the terrain and the wavy underwater effect—it involved a lot of learning HLSL and the many ways of using shaders. I truly believe the shaders were crucial in creating the atmosphere and setting I was aiming for. I'm grateful that this project has pushed me to learn so many new skills and grow as a developer.