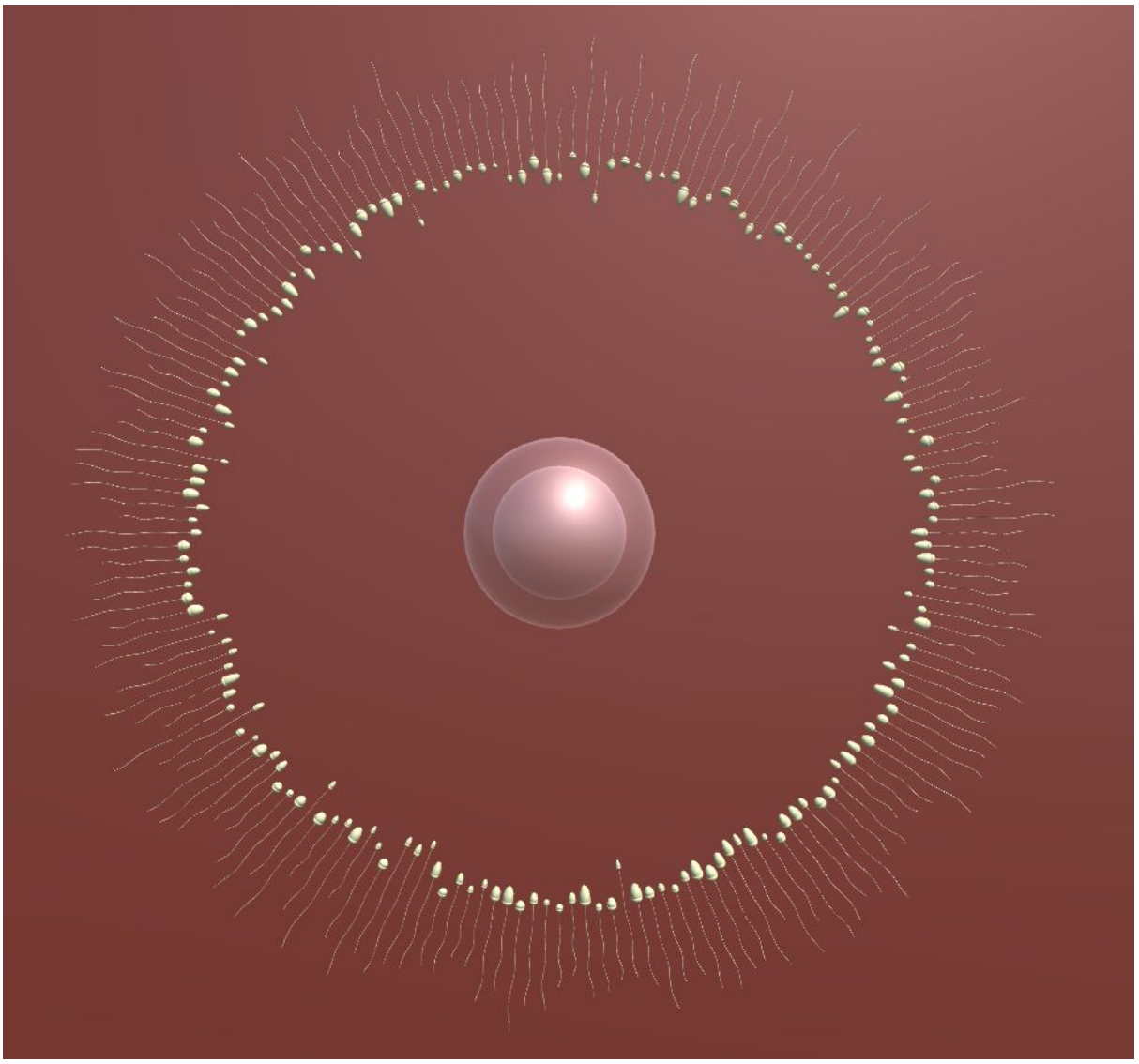
Algoritmos Genéticos (ConceptArt)

by Alfredo Pérez Pastor
Programación de Videojuegos 4.3



Índice

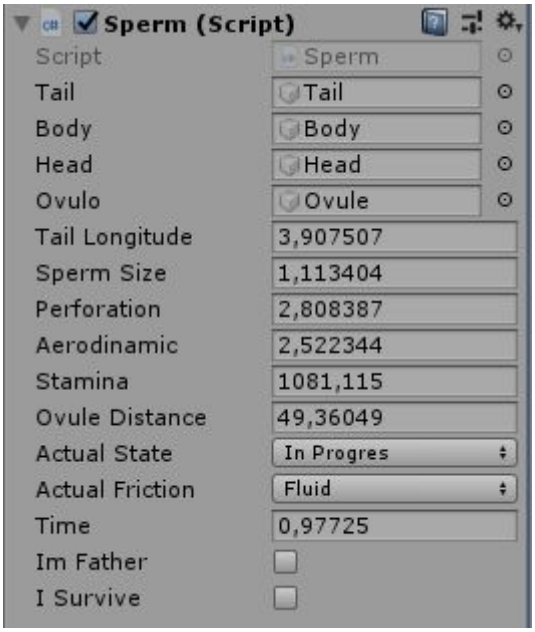
Introducción

En esta entrega, se realiza una implementación de Algoritmos Genéticos, para ello se ha tomado un sistema celular para ver cómo afectan diferentes atributos a los espermatozoides en su desplazamiento, en la velocidad y la eficiencia para fecundar un óvulo.

Implementación

Se han creado dos clases principales, una Sperm, que tiene como Cromosomas la longitud de la cola, el tamaño de la cabeza, y la capacidad de perforación de la punta. Estos valores generan otros, como la Aerodinámica y la Stamina, que serán claves para ver la forma en que se desplaza por los distintos medios.

Y la clase Genetic Algorithm que tiene se encarga de generar espermatozoides suficientes, sus atributos, modificación y herencia.



Para verlo las clases de manera más detallada estas son las funciones y todos los atributos de las clases.

Sperm

```
public class Sperm : MonoBehaviour {

    public GameObject tail;
    public GameObject body;
    public GameObject head;
    public GameObject ovulo;
    Rigidbody2D rigBody;

    float tailRotation = 0;
    float tailSpeed = 1000f;

    // Cromosomas
    public float tailLongitude = 3f;
    public float spermSize = 1f;
    public float perforation = 2f;

    // Derivate Values
    public float aerodinamic;
    float spermSpeed;
    public float stamina;

    // Movement States
    public enum CharacterState { inProgres, Death, Win };
    public CharacterState actualState = CharacterState.inProgres;

    public enum AmbientFriction { Fluid = 100, OvuleWall = 3 }
    public AmbientFriction actualFriction;

    static float ovuleOutDistance = 13.5f;
    static float ovuleInnerDistance = 8.5f;

    //AlgorithmValues
    public float ovuleDistance;
    public float time;
    public bool iSurvive;
```

```
// Start is called before the first frame update
void Start()...

// Update is called once per frame
void Update()...

// Control Manual del Esperma
void ManualControll() ...

// Control de la maquina de estados
void StateControll() ...

// Orientacion Constante al Ovulo, tanto para el impulso como para visualizarlo de forma correcta
void ShowToAim(Transform aim) ...

// Inicializa los valores principales para reiniciar la partida y recalcula los valores derivados.
public void RestartAtributtesRelation() ...

// Impulso frontal del esperma
void MoveForward() ...

// Movimiento lateral para el control manual
void RightMovement() ...
void LeftMovement() ...

// Calculo del la frenada que ejercen los fluidos del ambiente, y cambio de fluidos o densidades.
void FluidFriction() ...

// Raycast del impulso del esperma
void RaycastDraw() ...

// Movimiento del Flagelo en funcion del desplazamiento.
void TailMovement() ...

// Reestablece la apariencia del esperma
public void RestartCharacter() ...

// Cambio de material para los cambios generacionales
public void ChangeHeadMaterial(Material mat) ...
public void ChangeBodyMaterial(Material mat) ...
public void ChangeFullMaterial(Material mat) ...
```

Genetic Algorithm

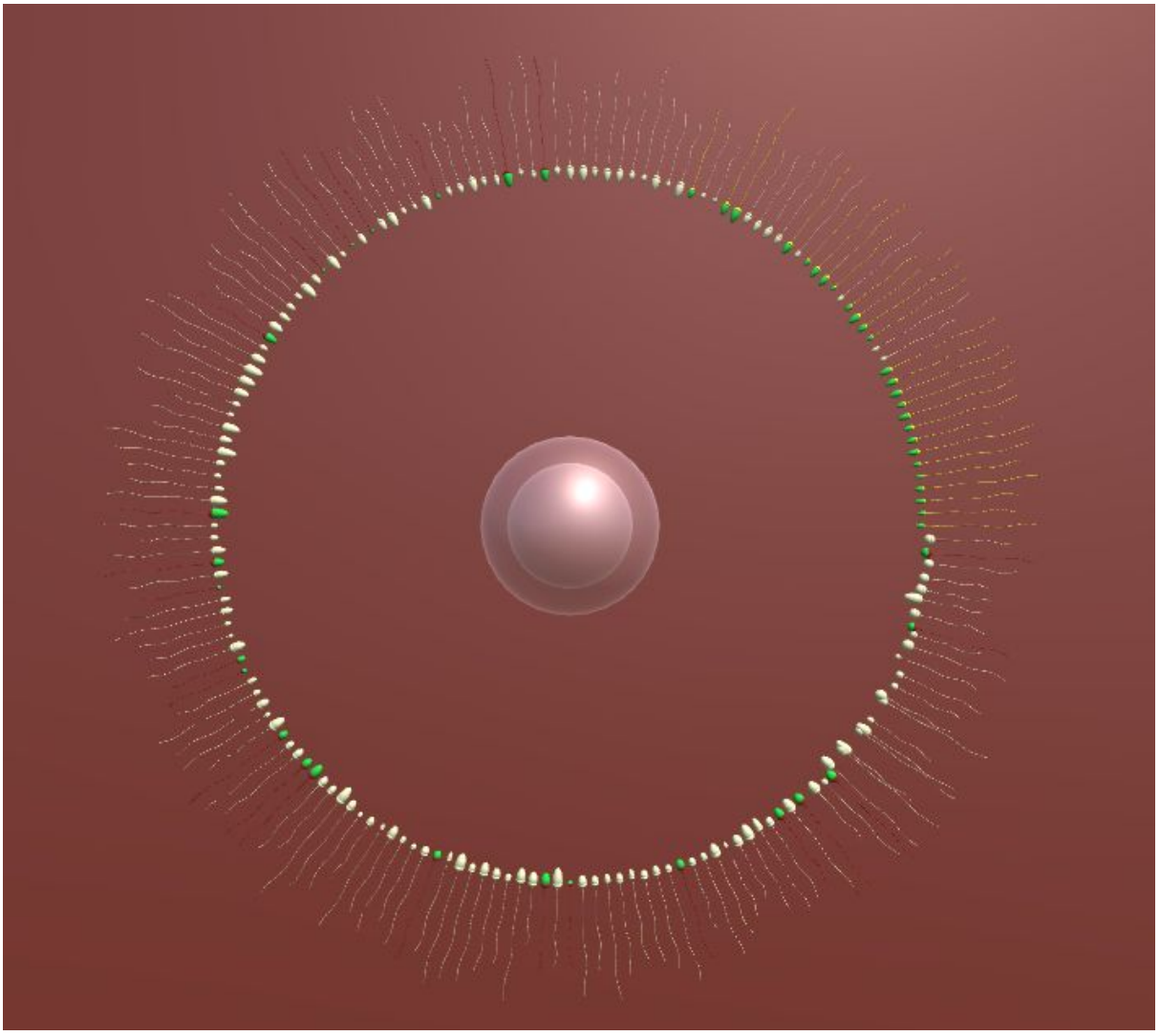
```
public class GeneticAlgorithm : MonoBehaviour
{

    public GameObject sperm;
    GameObject[] spermPull;
    public int pullSize = 90;
    public int generationSurvivors = 50;
    int actualSurvivors = 0;
    public float speedSimulation = 5;

    public float mutationProbability = 0.1f;

    float spermDistance = 50;
    float angleSpermDistribution;

    enum AlgorithmState { Inizialice, GenerationRun, Selection, Finish };
    AlgorithmState algorithmState = AlgorithmState.Inizialice;

    public Material mSperm;
    public Material mSpermChampion;
    public Material mSpermSurvive;
    public Material mSpermDeath;

    //UI Info
    [HideInInspector]
    public int generation = 1;
    [HideInInspector]
    public float winners = 0;
    [HideInInspector]
    public float bestTime = 0;
```

```
// Start is called before the first frame update
void Start()...

// Genera la Pull de Esperma y Crea Cromosomas
void GenerateFirstGeneration() ...

// Randomiza los Cromosomas de Un Esperma
void AsignRandomSpermValues(GameObject sperm) ...

// Coloca el esperma en forma de Helice, para ver el ranking.
void SpermHelixRecolocation() ...

// Coloca el esperma en un circulo para iniciar la generacion.
void SpermCicleRecolocation() ...

// Pinta y diferencia los que llegaron a la meta y los que no
void PaintWinnersAndLosers() ...

// Reordena todos los espermas en funcion del tiempo de fecundadion o de la
// maxima distancia de llegada en caso de no fecundar.
void ReorderByClassification() ...

// Consulta si todos los espermatozoides llegaron o murieron
bool isGenerationFinish() ...

// Devuelve los tiempos de todos los espermas
void PrintScores() ...

// Selecciona los mejores de forma que los mejores tienen mas posibilidades de seguir en la NextGen.
void RandomSelection() ...

// Reinicia los valores del esperma y los recoloca.
void StartGenerationRun() ...

// Crea un Nuevo Espermatozode en funcion de dos padres, con mezcla y probabilidad de mutacion.
void CreateSperm(GameObject father1, GameObject father2, GameObject child) ...

// Crea una generacion Nueva, mezclando los espermatozoides mas aptos.
void CreateNextGen() ...

// Reinicia los Atributos de toda la pull
void RestartPullAtributtes() ...

// Reinicia algunos parametros que va a tomar el HUD.
void RefreshInfo() ...

// Update is called once per frame
void Update()...
```

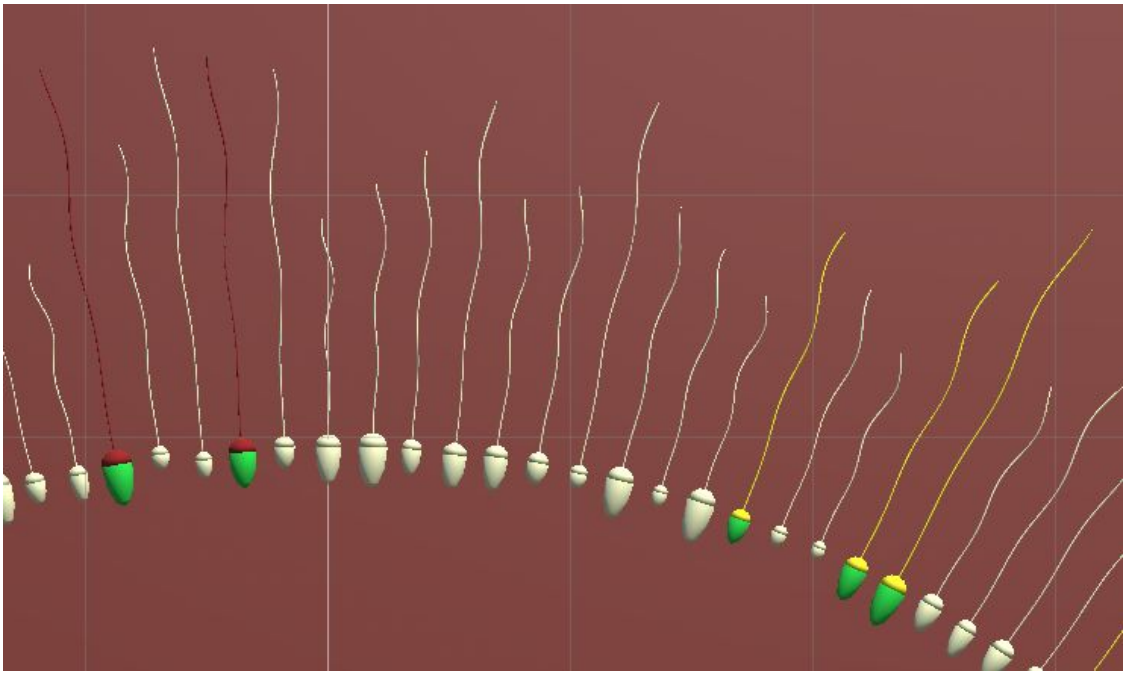
Visualización

Ordenación

Los espermatozoides más aptos se ponen desde la posición de 0º en sentido antihorario, y se toman con más probabilidad estos valores que los que están a 360º, que son los menos aptos.



Colores



Al crear el esperma pueden aparecer distintos colores, los cuales se detallan a continuación:

- Cabezas Verdes: seleccionados de la última generación.
- Flagelo Amarillo: espermatozoides de la última generación que lograron fecundar.
- Flagelo Rojo: espermatozoides de la última generación que NO lograron fecundar.
- Espermatozoides Blancos: nueva generación aún por valorar.

Conclusión

Esta entrega ha sido la más amena de todas, principalmente porque el tema da mucha rienda a la imaginación, sino también por la satisfacción que da como ves que en cada generación van mejorando poco a poco. En mi caso el mejor llevaba a la meta en 7.5 segundos.