Proyecto Algoritmos

La gran Colombia

Documentación

2025

Roles de los Integrantes

- **Santiago Diaz**: Gerente.
- **Gabriela Morales**: Diseño de juego y Algoritmos.
- **Danna Gonzales**: Desarrollador de backend y estructura de datos.
- **Valentina Mesa**: Doumentación y pruebas.

Contents

1 Introducción

"La Gran Colombia" es un prototipo académico de juego de estrategia por turnos 1 vs 1, inspirado en Risk y Supremacy 1914, ambientado en el mapa histórico de la Gran Colombia (Colombia, Venezuela, Ecuador y Panamá, 1821-1831). El tablero inicial está compuesto por 12 provincias interconectadas, suficiente para mostrar mecánicas de conquista sin sobrecargar al jugador. En cada ronda, los jugadores rivales planifican movimientos y ataques; luego todas las acciones se resuelven simultáneamente antes de pasar al siguiente turno.

Metas del prototipo v0.1

1. **Mapa jugable** con las 12 provincias clicables y resaltado al pasar el cursor.
2. **Sistema básico de tropas y recursos** dinero y unidades de infantería.
3. **Condición de victoria clara** controlar 8 de las 12 provincias.
4. **Interfaz minima** panel de información de provincia, botones "Mover" y "Atacar", contador de turnos.

2 Requisitos

Para ejecutar este proyecto localmente se necesita:

- **Sistema operativo:** Windows 10/11, macOS 10.13+ o Linux (Ubuntu 18.04+)
- **Unity** (versión recomendada: 2021.3 LTS o superior)
- **Procesador:** Intel Core i3 o equivalente.
- **Memoria RAM**: 4 GB mínimo.
- **Espacio en disco**: 1 GB libre.
- **Tarjeta gráfica**: Compatible con OpenGL 3.2 o superior.
- **Paquete TextMeshPro** (se instala automáticamente al abrir el proyecto en Unity si falta).

2.1 Instalación

1. **Clona o descarga el repositorio:**

```
git clone
   https://github.com/usuario/AlgoritmosProyecto.git
```

2. **Abre el proyecto en Unity:**
   - Abre Unity Hub.
   - Haz clic en "Add" y selecciona la carpeta del proyecto descargado.

3. **Instala dependencias si es necesario:**
    - Al abrir el proyecto, Unity puede pedirte instalar TextMeshPro u otros paquetes. Acepta e instala.
4. **Ejecuta el juego**
    - Selecciona la escena principal (por ejemplo, MenuInicial o la escena de inicio).
    - Haz clic en el botón **Play** en el editor de Unity para probar el juego localmente.

---

3 Manual de Usuario

Menú Principal

- Al iniciar el juego, verás el menú principal con las opciones Jugar y Salir.
- Pulsa Jugar para pasar a la selección de país.
- Pulsa Salir para cerrar el juego.

Selección de País

- Elige el país para cada jugador usando los menús desplegables.
- No es posible seleccionar el mismo país para ambos jugadores.
- Pulsa **Confirmar** para iniciar la partida con las selecciones realizadas.

Juego

- El mapa muestra las provincias disponibles. Cada jugador controla un conjunto de provincias iniciales.
- El juego es por turnos. En cada turno, el jugador puede construir edificios, reclutar tropas y mover unidades.
- Accede al menú de construcción seleccionando una provincia.
- Construye edificios si tienes recursos suficientes.
- Recluta tropas en provincias con los edificios adecuados.
- Ataca provincias adyacentes controladas por el enemigo.
- El resultado de la batalla depende de las tropas y edificios defensivos.
- El juego termina cuando un jugador controla todas las provincias o cumple condiciones de victoria específicas.

---

4 Guía Interna del Código

4.1 Resumen de las funciones principales

- **Simulación de Batallas:** `Ejercito.ResolverBatalla` calcula el resultado de los combates entre ejércitos, considerando ataque, defensa y efectos de edificios.
- **Administración de recursos:** `RecursosJugador` gestiona los recursos del jugador, verifica pagos y produce recursos por turno.
- **Construcción y producción:** `Departamento` y `Edificios` gestionan la construcción de edificios y la producción de tropas y recursos.
- **Gestión de turnos:** `SistemaDeTurnos` controla el flujo de turnos y la producción automática de recursos y construcciones.
- **Menús y navegación:** Scripts como:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class MenuInicialScript : MonoBehaviour
{
    public GameObject PanelMenuInicial;   // Panel con botones Jugar y Salir
    public GameObject SeleccionPaisPanel;   // Panel de seleccion de pais

    void Start()
    {
        PanelMenuInicial.SetActive(true);     // Mostrar menu principal
        SeleccionPaisPanel.SetActive(false);    // Ocultar seleccion pais
    }

    public void Jugar()
    {
        PanelMenuInicial.SetActive(false);    // Oculta el menu principal
        SeleccionPaisPanel.SetActive(true);
    }

    public void Salir()
    {
        Debug.Log("Saliendo...");
        Application.Quit();
    }

}
```

Listing 1: Clase Menú inicial

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;

public class SeleccionPaisController : MonoBehaviour
{
    public TMP_Dropdown dropdownJugador1;
    public TMP_Dropdown dropdownJugador2;
    public Button botonConfirmar;

    private List<string> opcionesOriginales = new
       List<string> { "Nueva Granada", "Ecuador",
       "Venezuela" };

    void Start()
    {
        // Carga las opciones en los dropdowns
        dropdownJugador1.ClearOptions();
        dropdownJugador1.AddOptions(opcionesOriginales);

        dropdownJugador2.ClearOptions();
        dropdownJugador2.AddOptions(opcionesOriginales);

        // Cuando cambia la seleccion de jugador 1,
           actualiza opciones de jugador 2
        dropdownJugador1.onValueChanged.AddListener(OnJugador1Cambio

        // Controla el boton Confirmar: solo habilitalo
           si paises son diferentes
        botonConfirmar.interactable = false;
        dropdownJugador1.onValueChanged.AddListener(delegate
           { ValidarConfirmar(); });
        dropdownJugador2.onValueChanged.AddListener(delegate
           { ValidarConfirmar(); });
        }

    void OnJugador1Cambio(int index)
    {
        string paisSeleccionado =
           opcionesOriginales[index];

        // Prepara opciones para jugador 2 sin el pais
           que escogio jugador 1
        List<string> opcionesJugador2 = new
           List<string>(opcionesOriginales);
        opcionesJugador2.Remove(paisSeleccionado);

        // Guarda seleccion previa de jugador 2 (si
           existe)
```

```
41              string seleccionActualJugador2 =
                    dropdownJugador2.options[dropdownJugador2.value].text;
42
43              // Actualiza las opciones de jugador 2
44              dropdownJugador2.ClearOptions();
45              dropdownJugador2.AddOptions(opcionesJugador2);
46
47              // Si la seleccion previa sigue valida,
                    mantenla, si no selecciona la primera opcion
48              int nuevoIndex =
                    opcionesJugador2.IndexOf(seleccionActualJugador2);
49              if (nuevoIndex >= 0)
50                  dropdownJugador2.value = nuevoIndex;
51              else
52                  dropdownJugador2.value = 0;
53
54              dropdownJugador2.RefreshShownValue();
55
56              ValidarConfirmar();
57          }
58
59          void ValidarConfirmar()
60          {
61              // Habilita el boton Confirmar solo si las
                    selecciones son diferentes
62              botonConfirmar.interactable =
                    dropdownJugador1.value !=
                    dropdownJugador2.value;
63          }
64
65          public void ConfirmarSeleccion()
66           {
67                  var seleccion = ObtenerPaisesSeleccionados();
68                  Debug.Log("Jugador 1 eligio: " +
                        seleccion.Item1);
69                  Debug.Log("Jugador 2 eligio: " +
                        seleccion.Item2);
70
71                  // Aqui continua la logica para empezar el
                        juego con esos paises.
72                  // Por ejemplo: ocultar el panel y activar
                        el mapa, cargar datos, etc.
73              }
74
75
76              public (string, string)
                    ObtenerPaisesSeleccionados()
77              {
78                  string pais1 =
                        opcionesOriginales[dropdownJugador1.value];
79                  List<string> opcionesJugador2 = new
```

```
                    List<string>(opcionesOriginales);
80                  opcionesJugador2.Remove(pais1);
81                  string pais2 =
                        opcionesJugador2[dropdownJugador2.value];
82                  return (pais1, pais2);
83              }
84          }
```

Listing 2: SeleccionPaisController

```
1           using System.Collections;
2           using System.Collections.Generic;
3           using UnityEngine;
4
5           public class BotonJugarySalir : MonoBehaviour
6           {
7               // Start is called before the first frame update
8               void Start()
9               {
10
11              }
12
13              // Update is called once per frame
14              void Update()
15              {
16
17              }
18          }
```

Listing 3: BotonJugarySalir

4.2 Estructura del código

Assets/scripts

- `RecursosJugador.cs` Lógica de recursos del jugador.
- `classJugador.cs` Representa a cada jugador.
- `Departamento.cs` Provincias del mapa.
- `Edificios.cs` Tipos de edificios y sus efectos.
- `classTropas.cs` Definición de tropas.
- `Ejercito.cs` Gestión de ejércitos y batallas.
- `SistemaDeTurnos.cs` Control de turnos.
- `MapaControlador.cs` Inicialización y conexiones del mapa.

- `MenuDepartamentoUI.cs, MenuConstruccionUI.cs` Interfaces de usuario para provincias y construcción.
- `MenuInicialScript.cs` Controla el menú principal y la transición a la selección de país.
- `SeleccionPaisController.cs` Lógica para la selección de países por los jugadores.

4.3 Tecnologías utilizadas

- **Unity Engine**: Motor principal del juego.
- **C**: Lenguaje de programación.
- **TextMeshPro**: Para la interfaz gráfica.
- **Sprites y Audio**: Para la representación visual y sonora.

4.4 Componentes clave

- **RecursosJugador**: Controla los recursos (dinero, comida, hierro, etc.), pagos y producción.
- **Departamento**: Nodo del grafo, representa una provincia. Gestiona edificios, tropas y producción.
- **Edificios**: Clases para cada tipo de edificio (Cuartel, Establo, Fábrica, Fortaleza), con sus efectos y costos.
- **Tropa y Ejercito**: Definen las unidades militares y su agrupación para batallas.
- **SistemaDeTurnos**: Controla el flujo de juego, alternando entre jugadores y procesando producción/construcción.
- **MapaControlador**: Inicializa el grafo de provincias y sus conexiones.
- **UI (MenuDepartamentoUI, MenuConstruccionUI)**: Interfaz para mostrar información y permitir acciones al usuario.
- **Menús principales**:
  - **MenuInicialScript**: Muestra el menú principal y gestiona la transición a la selección de país.
  - **SeleccionPaisController**: Permite a los jugadores elegir sus países y valida la selección.
  - **BotonJugarySalir, ScriptJugar**: Scripts para botones de navegación y acciones básicas.

4.5 Interacciones Entre Módulos

- **Jugador** contiene un RecursosJugador y un Ejercito
- **Departamento** referencia a su Dueño (Jugador) y contiene listas de Edificios y Tropas.
- **SistemaDeTurnos** itera sobre los jugadores y sus departamentos para procesar producción y construcciones.
- **UI** interactúa con los scripts de lógica para mostrar información y ejecutar acciones (construir, reclutar, atacar).
- **Menús** gestionan el flujo inicial del juego y la configuración de la partida.

5 Comparación de Avances Semanales

- **Semana 1 - Investigación:** Se definieron las mecánicas básicas del sistema de combate. Se analizaron juegos de estrategia como referencia.
- **Semana 2 - Planeación:** Inició la creación del documento de diseño del videojuego. Se definieron los objetivos generales del proyecto y su estructura inicial. **Problema:** Se está tardando más de lo esperado en consolidar toda la información en el documento.
- **Semana 3 - Diseño del mapa:** Se completó el boceto del mapa con divisiones políticas por departamentos. El diseño se basó en referentes históricos y geográficos.
- **Semana 4 - Backend:** Se desarrollaron clases clave para el funcionamiento del juego (provincias, jugadores, recursos). Se inició la conexión con una base de datos local para gestionar información persistente. **Problema:** Algunas clases aún no están completamente integradas entre sí.
- **Semana 5 - Pantalla principal:** Se completó la pantalla inicial del juego con navegación básica. La estética visual fue alineada con la temática histórica del juego.
- **Semana 6 - Menús e implementación de turnos:** Se crearon menús interactivos para cada provincia. Se implementó el sistema de turnos como base del flujo del juego.
- **Semana 7 - Sistema de combate y recursos:** Se ajustaron las reglas de combate para facilitar la jugabilidad. Se finalizó el menú principal con diseño funcional. Se añadió la lógica para asignar recursos a cada departamento.