Carátula de Proyecto – Taller Flappy Bird

Título del proyecto: Flappy Bird
Alumno/a: Miguel Carmona Garrido

Objetivo

Implementar un **Flappy Bird** funcional con los **requisitos mínimos** que se listan a continuación. Esta carátula se entrega junto al proyecto y sirve como checklist de cumplimiento.

Requisitos mínimos (Checklist) — Unity

Área	Requisito mínimo (Unity)	Cumple
Motor	**Unity 2021 LTS o superior**; proyecto 3D	[x]
Lenguaje	**C#** con scripts MonoBehaviour; nombres claros	[x]
Control	**Input único** para salto (tecla **Espacio**, **clic** o **toque**). Puedes usar **New Input System** (Action "Jump") o Input.GetKeyDown(KeyCode.Space)	[x]
Física	Rigidbody en el pájaro (**Gravity Scale** > 0) y **impulso vertical** con AddForce o set de velocity.y	[x]
Colisiones	Collider (pájaro: SphereCollider; tuberías/suelo: BoxCollider). **Layers** y Collision Matrix configurados. On collision → **Game Over**	[x]
Obstáculos	**Spawner** con **Object Pooling** que instancie **parejas de tuberías** con hueco aleatorio, desplazamiento constante (mover por Rigidbody o Transform.Translate)	[x]
Scroll	Movimiento lateral de tuberías/terreno; **parallax** opcional con varias capas SpriteRenderer	[x]
Puntuación	+1 al **cruzar el trigger** entre tuberías (OnTriggerEnter). **UI** con puntuación actual y **High Score**	[x]
Estados	**GameManager** con estados: **Ready → Playing → GameOver**; botón **Retry** que recargue escena	[x]
Escenas	1 escena de juego (Game.unity) + overlay/panel de inicio/fin (u otra escena simple de título)	[x]

Área	Requisito mínimo (Unity)	Cumple
UI mínima	**TextMeshPro** para Score, High Score, Start, Retry (Canvas en **Screen Space - Overlay**)	[x]
Audio	2 **AudioClips** (salto/colisión) con AudioSource y volúmenes equilibrados	[x]
Arte mínimo	**Prefab** del pájaro (2–3 frames con Animator o rotación por velocidad), **tubería**, **fondo**	[x]
Rendimiento	Objetivo **60 FPS** PC / **30 FPS** móvil. Configurar Application.targetFrameRate, **VSync desactivado** en Calidad si aplica	[ ]
Resolución	Diseño 16:9 (p. ej. 1920×1080 / 1280×720).	[x]
Build	**Windows** en Build Settings; probar build final	[x]
Persistencia	**PlayerPrefs** para guardar High Score con clave estable (p. ej. "HIGH_SCORE") o Binary Formatter para un mejor y correcto uso	[x]
Código	Scripts clave: BirdController, PipeController, PipeSpawner?, ScoreZone?, GameManager, UIController	[x]
Estructura	Carpetas: Assets/Art, Assets/Audio, Assets/Prefabs, Assets/Scenes, Assets/Scripts, Assets/Fonts	[x]
Control de versiones	Proyecto en **Git** con **.gitignore de Unity** (Library/, Logs/, Temp/, Obj/, Build/…); **README.md**	[x]
Licencias	Citar fuentes/licencias de arte y audio (CC0/CC-BY, etc.)	[x]

Entregables (Unity)

- **Build**: ejecutable **Windows (.exe + _Data)**.
- **Proyecto Unity** completo (Assets, Project Settings y Packages)
- **README.md** con: versión de Unity, controles, cómo construir/ejecutar, y créditos/licencias.
- **Carátula (este documento)** con el checklist marcado.

Criterios de evaluación básicos (Unity)

Criterio	Descripción	Peso
Jugabilidad	Control de salto consistente; huecos y velocidad ofrecen reto progresivo	30%
Estabilidad/Física	Colisiones fiables; sin atascos;	25%
Claridad/UI	Puntuación visible con **TextMeshPro**; mensajes de estado y feedback sonoro claros	20%
Organización	Scripts y **Prefabs** ordenados; carpetas Unity estándar; .gitignore correcto; README completo	15%
Build/Perf	Build funcional (WebGL/Win/Android); FPS objetivo alcanzado; tamaño razonable	10%

Pistas técnicas rápidas

- Impulso: rb.velocity = new Vector3(x, jumpSpeed, z); o rb.AddForce(Vector3.up * force, ForceMode.Impulse);
- Puntuación: OnTriggerEnter(Collider other) { if (other.gameobject.tag == "Score") score++; }
- Game Over: desactivar spawner y movimiento; mostrar panel UI; permitir Retry con SceneManager.LoadScene.
- Guardado: PlayerPrefs.SetInt("HIGH_SCORE", Mathf.Max(actual, PlayerPrefs.GetInt("HIGH_SCORE", 0)));