

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

Área	Requisito mínimo (Unity)	Cumple
UI mínima	<b>TextMeshPro</b> para Score, High Score, Start, Retry (Canvas en <b>Screen Space - Overlay</b> )	[ ]
Audio	2 <b>AudioClips</b> (salto/colisión) con AudioSource y volúmenes equilibrados	[ ]
Arte mínimo	<b>Prefab</b> del pájaro (2–3 frames con Animator o rotación por velocidad), <b>tubería, fondo</b>	[ ]
Rendimiento	Objetivo <b>60 FPS</b> PC / <b>30 FPS</b> móvil. Configurar Application.targetFrameRate, <b>VSync desactivado</b> en Calidad si aplica	[ ]
Resolución	Diseño 16:9 (p. ej. 1920×1080 / 1280×720).	[x]
Build	<b>Windows</b> en Build Settings; probar build final	[ ]
Persistencia	<b>PlayerPrefs</b> para guardar High Score con clave estable (p. ej. "HIGH_SCORE") o Binary Formatter para un mejor y correcto uso	[ ]
Código	Scripts clave: BirdController, PipeController, PipeSpawner?, ScoreZone?, GameManager, UIController	[ ]
Estructura	Carpetas: Assets/Art, Assets/Audio, Assets/Prefabs, Assets/Scenes, Assets/Scripts, Assets/Fonts	[ ]
Control de versiones	Proyecto en <b>Git</b> con <b>.gitignore de Unity</b> (Library/, Logs/, Temp/, Obj/, Build/...); <b>README.md</b>	[ ]
Licencias	Citar fuentes/licencias de arte y audio (CC0/CC-BY, etc.) <a href="https://megavtogal.com/52-foto/flappy-bird-png-fon.html?utm_referrer=https%3A%2F%2Fwww.bing.com%2F">https://megavtogal.com/52-foto/flappy-bird-png-fon.html?utm_referrer=https%3A%2F%2Fwww.bing.com%2F</a>	[ ]

### 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 <b>TextMeshPro</b> ; mensajes de estado y feedback sonoro claros	20%
Organización	Scripts y <b>Prefabs</b> 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)));`