Open Opened 3 weeks ago by **Rubén Montero**

Crear tu propio juego

Resumen

- Presentaremos los requisitos para un nuevo juego
- Diseñaremos e implementaremos ese juego nosotros mismos, sin más guía que la documentación de Godot

Descripción

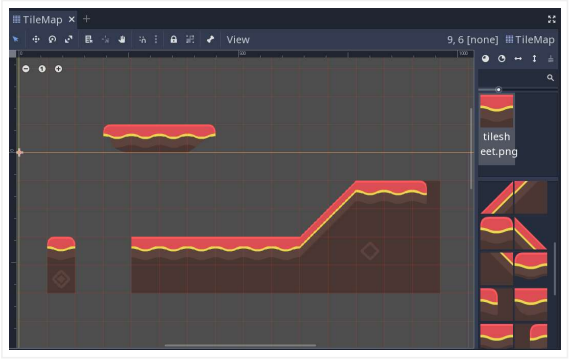
Vamos a diseñar e implementar nuestro propio videojuego. Esta es la **única** tarea de este *sprint*.

Pero... ¿Cómo?

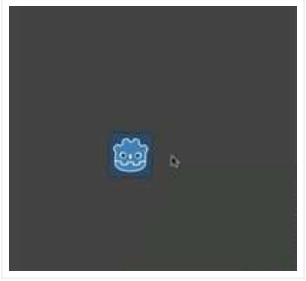
Piensa en una idea. Dale vueltas. Refínala, sin prisa.

Consulta los siguientes enlaces para hacerte una idea de cómo podrías implementar (a nivel técnico) algunas cosas:

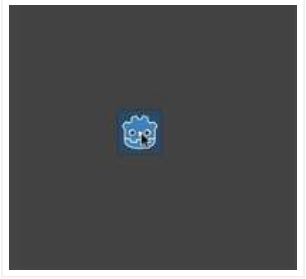
- Tilemaps: Utilidad que permite crear mapas 2D a partir de imágenes simples con *casillas* (*tilemaps*) especialmente diseñadas para encajar unas con otras y flexibilizar el diseño de un mapa de plataformas o de vista aérea, normalmente. Puedes encontrar recursos de licencia abierta en OpenGameArt.org



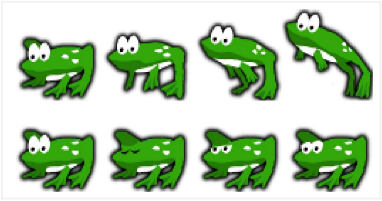
- Rotación y movimiento (ratón)



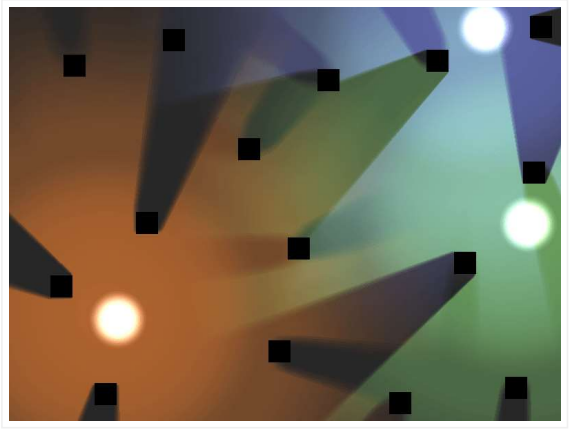
- *Click* y mover



- *Spritesheet*: Animación de personajes. Puedes encontrarse recursos de licencia abierta en OpenGameArt.org



- Luces y sombras en 2D



- Otros recursos y tutoriales

Pero... ¡tengo miedo! ¿Cómo empiezo?

Si aún no ves claro *cómo* puedes comenzar sobre un lienzo en blanco, puedes trabajar en las siguientes tareas *opcionales* que también están publicadas este *sprint*. ¡Ojo! No son evaluables. Todo lo que se evaluará será *este juego*.

Puedes orientarte por los criterios de evaluación más abajo.

Criterios de evaluación

Se usará la siguiente *rúbrica* de evaluación para obtener una nota numérica entre `0` y `20` de tu proyecto:

Criterios				
Personaje				
¿Cómo es el actor principal manejable por el usuario? ¿Qué acciones tiene permitidas?				
	3.0	2.0	1.0	0.0
	Complejidad elevada	Múltiples acciones especiales	Una acción especial	Actor simple
	El personaje principal puede moverse de forma multidireccional en al menos una dimensión, llevar a cabo más de una acción especial (e.g.: Saltar y Disparar, o Atacar y Bloquear) y es necesario emplear el ratón para algún control que requiera el procesamiento de coordenadas (no vale con usar los "clicks" del ratón para alguna acción que se podría hacer con el teclado), o bien el uso de la cámara conlleva una complejidad elevada (e.g.: Juego en primera persona, Vehículo), o bien hay múltiples actores manejables por el usuario que pueden ejecutar acciones distintas (e.g.: Juego de estrategia con diferentes tipos de unidades)	El personaje principal puede moverse de forma multidireccional en al menos una dimensión y llevar a cabo más de una acción especial (e.g.: Saltar y Disparar, o Atacar y Bloquear)	El personaje principal puede moverse de forma multidireccional en al menos una dimensión y llevar a cabo una acción especial (e.g.: Disparar o Saltar)	El personaje principal tiene un movimiento simple multidireccional (como en *Dodge the Creeps!*) o es controlable mediante una acción simple (e.g.: Sólo saltar)

Criterios				
Enemigos				
¿Cómo son los enemigos?				
	3.0	2.0	1.0	0.0
	Inteligentes	Varios tipos	Simples	Sin enemigos
	Enemigo(s) que lleva(n) a cabo acciones complejas, tratando de emular comportamiento humano mediante algoritmos de control elaborados	Dos ó más tipos de enemigo(s) claramente diferenciados que llevan a cabo acciones sencillas	Enemigo(s) que lleva(n) a cabo acciones sencillas (como en *Dodge the Creeps!*)	No hay enemigo(s) o bien son estáticos y su implementación no representa ningún tipo de dificultad
Complejidad de escenas				
¿Cómo de elaborado es el entorno en el que sucede la acción?				
	3.0	2.0	1.0	0.0
	Entorno transitable con interacciones y animaciones	Entorno transitable con interacciones	Entorno transitable	Entorno simple
	Todas las escenas jugables contienen áreas accesibles e inaccesibles (e.g.: Plataformas, Agua o Muro que el jugador no puede atravesar), elementos que permiten interacción del usuario (e.g.: Trampa que se activa, Puerta que puede abrirse) y elementos de la escena movibles a través de animaciones (e.g.: Plataforma móvil, Obstáculo)	Todas las escenas jugables contienen áreas accesibles e inaccesibles (e.g.: Plataformas, Agua o Muro que el jugador no puede atravesar) y elementos que permiten interacción del usuario (e.g.: Trampa que se activa, Puerta que puede abrirse)	Todas las escenas jugables contienen áreas accesibles e inaccesibles (e.g.: Plataformas, Agua o Muro que el jugador no puede atravesar)	Las escenas jugables no representan complejidad alguna (como en *Dodge the Creeps!*)
Numero de escenas				
¿Cuántos mapas o niveles claramente diferenciados existen?				
	3.0	2.0	1.0	0.0
	Tres o más	Dos	Uno	Cero
	Tres o más	Dos	Uno	Ninguno
Bonificaciones o coleccionables				
Además de enemigos, ¿hay elementos interactivos que son obtenibles por el actor principal?				
	2.0	1.0	0.0	
	Dos o más tipos	Un elemento simple	Sin elementos	

Criterios				
	Existen dos o más tipos de elementos interactivos que son recogibles por el usuario y, al menos uno, supone una ventaja significativa dentro del juego (e.g.: Vida extra y Moneda, Munición y Estrella)	Existe un elemento simple interactivo que es recogible por el usuario (e.g.: Moneda)	No hay elementos interactivos recogibles por el usuario	
Funciones de red				
¿El juego envía peticiones a un API REST?				
	3.0	2.0	1.0	0.0
	Dos ó más endpoints dentro de la escena del juego	Dos o más endpoints	Endpoint simple	Sin peticiones de red
	Se consumen dos ó más endpoints REST especialmente implementados para el juego usando otra tecnología (e.g.: Django, Restlet) y que tienen efecto visual para el jugador dentro de la escena(s) jugable(s). NO se plantea sincronización en tiempo real multijugador. Ello se implementaría con protocolos distintos a HTTP. El planteamiento es el uso de endpoints REST de manera síncrona en momentos específicos de la partida (e.g.: En un mapa los jugadores hacen POST de la ubicación donde pierden, y cuando juegas haces GET para pintar sus fantasmas)	Se consumen dos ó más endpoints REST que no tienen efecto visual dentro del juego. Puede utilizarse el API REST /score para publicar (POST) puntuaciones y obtener (GET) y mostrar las puntuaciones en un scoreboard	Se consumen un endpoint REST que no tiene efecto visual dentro del juego. Puede utilizarse el API REST /score para publicar (POST) puntuaciones	No hay uso del protocolo HTTP en el juego
Sonido				
¿Cómo es el sonido?				
	1.0	0.0		
	Elaborado	Poco elaborado o nulo		
	Existen dos ó más efectos de sonido simples que tienen lugar ante ciertos eventos del juego, y más de una música de fondo	Los recursos de audio empleados no exceden más de un efecto de sonido simple y/o música de fondo		
Opciones				
¿Se pueden configurar opciones que se persisten?				
	1.0	0.0		
	Hay opciones	Sin menú opciones significativo		
	Existe un menú de opciones que permite configurar dos ó más opciones (e.g.: Nombre de usuario y Dificultad, Volumen y Color del tema) que se persisten en almacenamiento secundario	No hay un menú de opciones significativo con más de una opción almacenada en disco		
Estética y jugabilidad				

Criterios				
¿Cuál es la calidad del producto final?				
	1.0	0.0		
	Buena calidad	Escasa calidad		
	La estética del proyecto y su jugabilidad se traducen en una buena calidad final	La estética o la jugabilidad exhiben problemas importantes y generan críticas justificables y relevantes que representan la escasa calidad final		

Requisitos obligatorios

IMPORTANTE

- El proyecto se debe crear desde cero. **No** debe ser un *fork* (una copia modificada) de otro proyecto existente.
- La historia en Git debe reflejar los avances en el desarrollo y **demostrar** que se trata de *tu desarrollo*. ¡No esperes más de 2 ó 3 días para hacer un nuevo *commit*!

No cumplir estos requisitos resultará en una **puntuación de** `0` (cero).

Por último

Enseña tu videojuego a compañeros y compañeras. ¿Qué opinan?

El *feedback* es muy importante en el proceso de mejora continuo.

Rubén Montero @ruben.montero changed milestone to %Sprint 4 3 weeks ago