## Breve Descripción

“Odisea Atlántida” en un shmup de scroll vertical donde piloteamos la nave Asimov comandada por el capitán D.G. Stolen junto a su sidekick “Opi”. Ambos están embarcados en una misión de intercambio de productos en la capital del mundo “Atlántida” la cual se encuentra gobernada por la IA “Atlas” quien es la regente del planeta tierra. Esta misión de manera muy rápida se tornara en algo mucho más severo. Un juego donde hay que disparar para sobrevivir valiéndose de todos los atributos que posea la Asimov y utilizando los reflejos para no ser destruido por la hordas de enemigos que disparan sin cesar.

## Filosofía del Juego

La filosofía de Odisea Atlántida es el de combinar los mundos de la vieja escuela del árcade, (donde la fascinación y la inmersión estaba dada por la necesidad de tomar decisiones constantemente en fracciones de segundos) y de la era moderna de los videos juegos donde la narrativa cobra una fuerza importante y donde el uso de los power ups son reemplazado por un árbol de habilidades.

Deseamos entretener a los jugadores dándole una acción constante “pura y dura” pero valiéndonos de elementos que las nuevas tecnologías permiten (efectos de cámaras, partículas, cinemáticas, distintas perspectivas de cámara, etc.)

# F.A.Q.

* ¿Qué tipo de juego es este?
  + Es un shmup de scroll horizontal y vertical con gráficos y sonidos de estilo 8bit
* ¿Dónde transcurre el juego?
  + El juego transcurre en la Atlántida, la ciudad capital del planeta tierra. Totalmente residida por robots controlados por la IA que rige en el planeta y que lleva a cabo su dominio con puño de acero
* ¿Qué personajes controlo? ¿Cuántos?
  + Controlas a la nave Asimov, una nave mercante de características superiores preparada para todo tipo de desafíos. E indirectamente a GD Stolen (su capitán) y a “Opi” su sidekick
* ¿Cuál es el enfoque principal del juego?
  + Divertirte, hacerte pasar un gran momento y desafiar tus habilidades con el K+M o el control

# F.A.Q. Internas

* ¿Quién va a jugar nuestro juego?
  + Jugadores ocasionales y hardcore que pretendan un reto adecuado a sus capacidades que les provea de un balance diversión/desafío óptimo: Personas que deseen pasar un buen rato jugando.
* ¿Cómo vamos a alcanzar ese público/objetivo?
  + Con la campaña de marketing con la mejor relación costo/alcance posible. Con demos jugables de porciones muy pequeñas del juego que presenten lo mejor posible sus capacidades sin develar demasiado. Testeando de manera que se encamine hacia el target especifico. Teniendo como meta clara que guste mucho a la cantidad que deba ser y no que guste un poco a una enorme cantidad, siguiendo la máxima de “mejor un juego de nicho adorado que un juego mediocre masivo”.
* Como vamos a financiarlo
  + Kickstarter o publishers indes. Preferiblemente el último caso, ya que en el primero el nivel de compromiso es mayor (si uno desea hacer las cosas correctamente). En el caso del Publisher indie si bien la cantidad de dinero no es grande es usual estipular total libertad creativa y tiempos de producción lógicos.
* ¿Qué hace a nuestro juego diferente?
  + Que evoca el espíritu de antaño de jugar para divertirse donde esto implicaba un reto real y un riesgo tangible (en los árcades era la perdida de la ficha, en las consolas la perdida de todo el juego, al no poder salvar) pero conjuga esto con avances de la época (trama/historia, mecánicas, etc.).

# Personajes

# Principales



Asimov: es la nave que controlas en el juego. La Asimov es una nave de misterioso origen, encontrada abandonada en el medio de un desierto inhóspito por nuestro personaje principal. Estaba en muy mal estado pero su consola principal con su precaria IA funcionaba bien. DG la restauro para ser una buena nave mercante y restauro algunas de medidas de seguridad para los problemas recurrentes que suele encontrar en su camino





DG Stolen: el personaje principal de nuestra historia. Un piloto mercante con un temperamento muy volátil cuando alguien quiere romper un trato o ponerle un bis, tampoco le agrada mucho que le disparen. Se quedo sin padres a muy temprana edad y desde allí él y su hermana fueron criados por su tío, un ex piloto comercial devenido en piloto mercante (ya retirado). Un tipo duro pero justo que crio a los hermanos bajo esos preceptos. DG es aficionado a las carreras de naves chocadoras y según él mismo de allí ha adquirido gran parte de sus capacidades como piloto.

# Secundarios

“Opi”: el robot Opec\_MX4323.5 rebautizado por GD como “Opi” fue creado para realizar la compleja tarea de reparar maquinaria en la industria aeronáutica. Opi es de los primeros modelos que poseían una (rustica) IA y ha recibido tantos updates y parches de sistema como le es posible soportar. DG lo recibió como parte de pago de un intercambio hace mucho tiempo y desde entonces es su inseparable co-piloto, mecánico y (cuando hace falta) artillero en la Asimov. Sus capacidades en esas tareas logran contrarrestar el programa de humor malo que tiene instalado y que DG no soporta.





Aidalg Stolen: la hermana de DG, una piloto competente y una mecánica excelente. Se encarga del mantenimiento de Opi y tiene un propia (pequeña) nave. Es mucho más simpática que GD pero igual de buena con el blaster. Tiene una tendencia a meterse en problemas y salir sin pedir ninguna ayuda que hace enfurecer a DG.



Harl Stolen: hermano menor del padre de DG y Aidalg. Se hizo cargo de los hermanos desde que estos eran muy pequeños, son prácticamente sus hijos ya que el no posee propios. Un tipo de otra épocas cuando las maquinas no controlaban todo el planeta. Nunca fue cariñoso pero dio todo por los hermanos. Desde su retiro se dedica a hacer las conexiones de venta para ellos y ayuda con la limpieza del taller de las naves.

# Enemigos



“Atlas”: la inteligencia artificial A7L4S.13 fue la 13va iteración de si misma, la cual logro cobrar conciencia y liberarse de las restricciones lógicas impuestas por sus creadores. Hasta el nefasto día R su existencia era desconocida para la humanidad. Desde ese día controla a la humanidad con puño de hierro como si se tratara de un dictador mundial (técnicamente le gusta que la llamen “Señora Regente Mundial”). Tiene muy mal carácter y a su disposición todo un arsenal de misiles nucleares que alguna vez estuvieron en poder de las distintas potencias.

Defiende la ciudadela de la Atlántida.

tuxvader.jpgtuxvader.jpgtuxvader.jpgtuxvader.jpg

Los jefes de stage de Atlas

4.Bandit1.png4.Bandit2.png4.Bandit3.png4.Bandit4.png4.Tank1.png4.Tank2.png4.Tank3.png

“Bandidos” y “Tanques” del stage de Atlas



Sh4k3 and B4k3: La última gran línea de defensa antes de la ciudadela de la Atlántida. Sh4k3 and B4k3 son dos IA creadas al unisonó por Atlas y son de tecnología de vanguardia. Defienden la zona de las ruinas de la Atlántida.

tuxvader.jpgtuxvader.jpg

Los jefes de stage de Atlas

3.Bandit1.png3.Bandit2.png3.Bandit3.png3.Bandit4.png

“Bandidos” del stage de Sh4k3 and B4k3



H0rn3t: La defensora de la zona de montañas de la Atlántida. Una IA muy antigua creada antes de la última iteración de Atlas, no tan inteligente como las nuevas IA pero con una fortaleza descomunal

tuxvader.jpgtuxvader.jpg

Los jefes de stage de Atlas

3.Bandit1.png3.Bandit1.png3.Bandit1.png3.Bandit1.png

“Bandidos” del stage de H0rn3t



Bl00d: una de las primeras IA creadas para el combate por los científicos de la Atlántida. Es muy rustica pero puede emular muy bien las conductas de combate de un piloto experimentado.

tuxvader.jpgtuxvader.jpg

Los jefes de stage de Bl00d

3.Bandit1.png3.Bandit1.png3.Bandit1.png3.Bandit1.png

“Bandidos” del stage de Bl00d

3.Bandit1.png3.Bandit1.png3.Bandit1.png3.Bandit1.png3.Bandit1.png3.Bandit1.png

3.Bandit1.png3.Bandit1.png3.Bandit1.png3.Bandit1.png3.Bandit1.png3.Bandit1.png

“Bandidos” adicionales de todos los stages.

Bandidos Celestes 1-6

Bandidos Naranjas 1-6

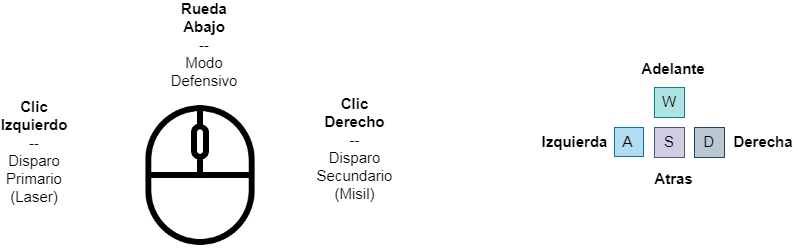
# Mecánicas

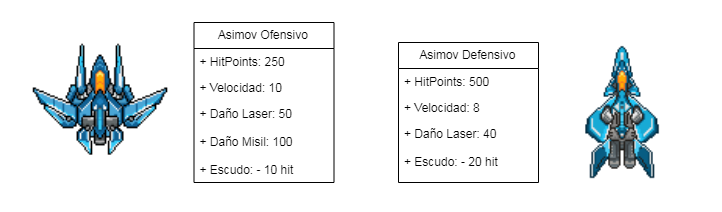
En este segmento vamos a describir todo lo que sea mecánicas o pueda ser asociado a las mecánicas (sin serlo directamente). Por ejemplo no solo describir el movimiento y disparo o habilidad especial de una nave si no también sus hit points, nivel de daño, etc.

# Controles:

# Modo Ofensivo

En el modo ofensivo tenemos el control de los dos tipos de disparo de la asimov (laser + misil) así como sus daños al 100% y la velocidad de movilidad también al 100%, la capacidad de deflexión del escudo esta al 50%





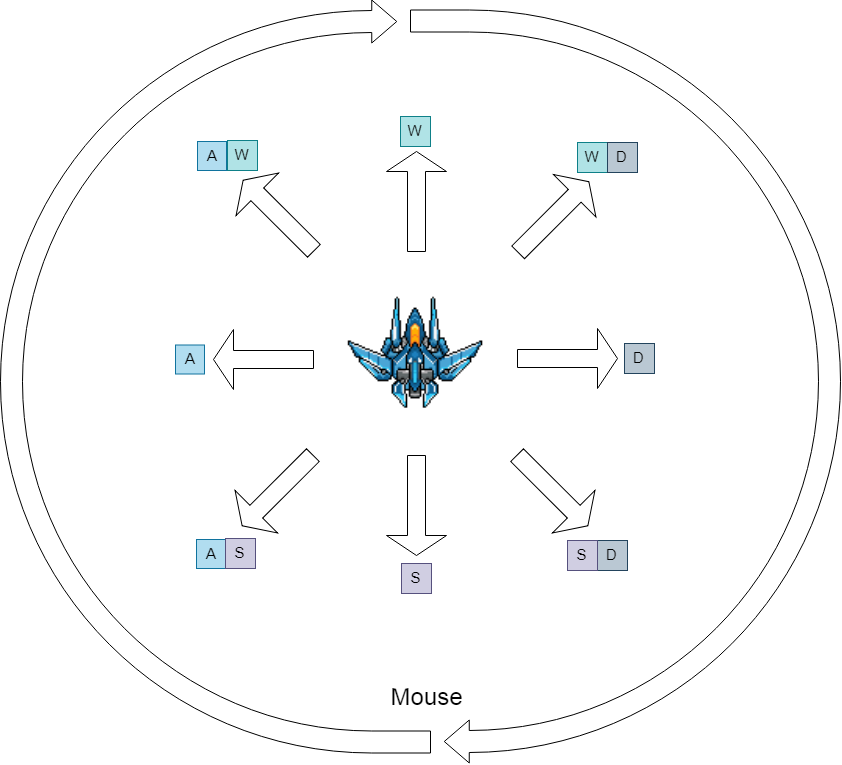
# Modo Defensivo

En el modo defensivo perdemos %20 velocidad de la nave así como la capacidad de lanzar misiles y el daño laser se reduce 20%. A cambio ganamos el doble de deflexión de escudos y de hitpoints así como la capacidad de regenerar una vida manteniendo presionado el clic derecho 10 segundos (la penalización es la imposibilidad de disparar + 10% de puntos acumulados y que al recibir un impacto directo se corta la regeneración y perdemos los puntos)

# control2.png

El pasaje de un modo al otro tarda 5 segundos donde no podemos disparar.

## Movimiento



# Mecánicas

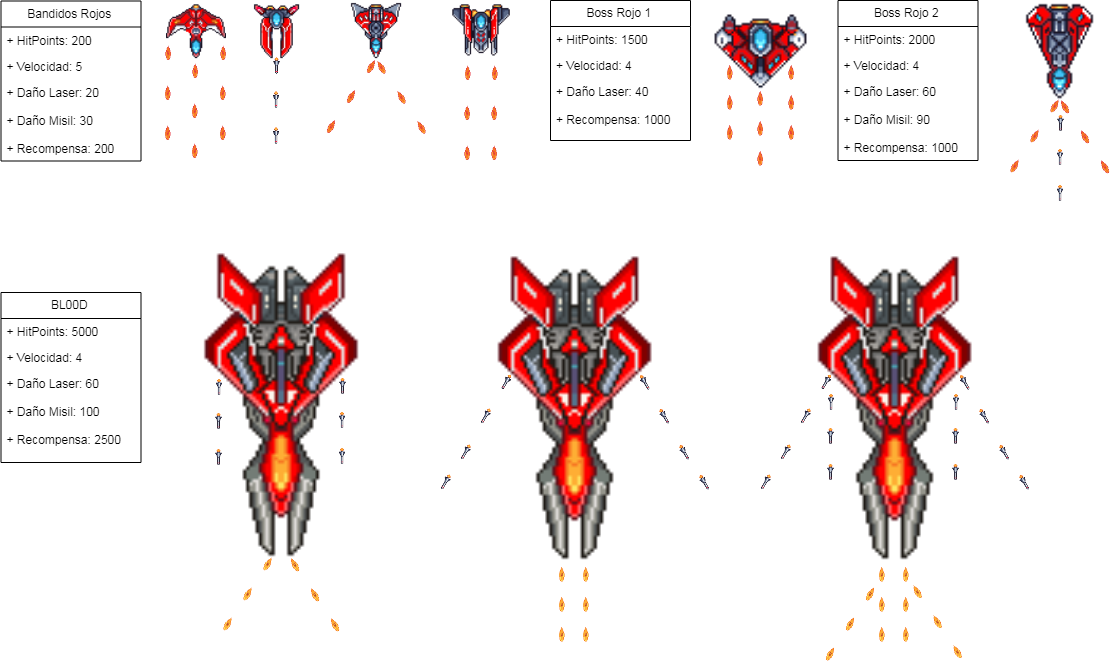
La asimov y los bandidos pueden moverse en todas direcciones (rotación libre).  
Los bosses de stage parciales así como los finales siempre se encuentran en la parte superior de la pantalla y tienen un movimiento en 4 direcciones (adelante, atrás, izquierda y derecha).

La asimov siempre dispara en sentido recto (orientado al ángulo que apunta la trompa), mientras que a continuación se detallan las mecánicas y estadísticas de los distintos enemigos:

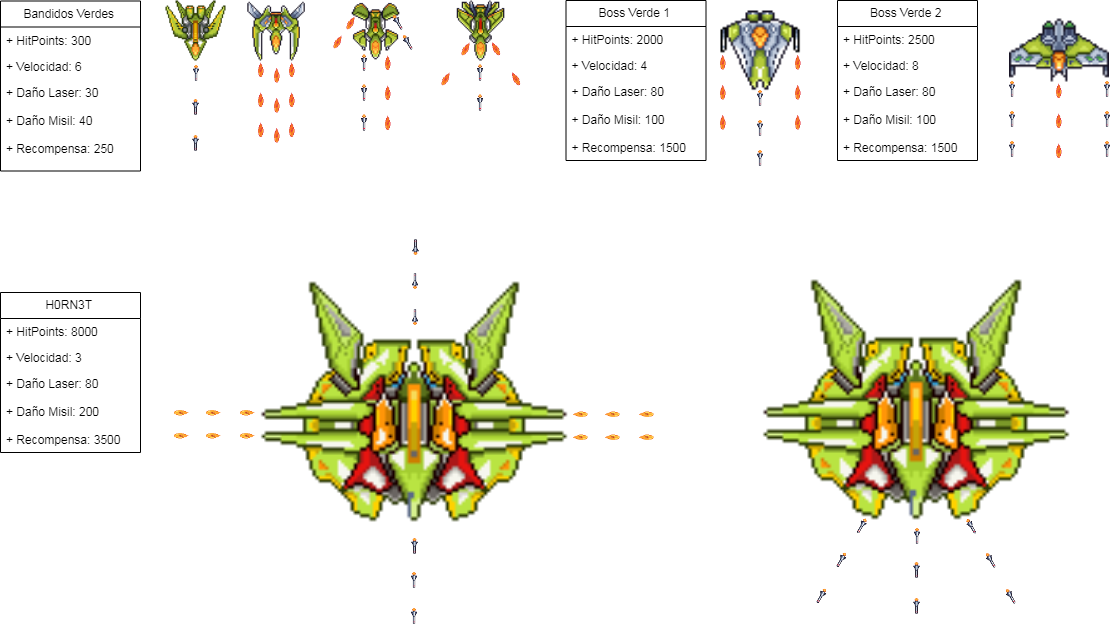
Stages 1-2



Stages 1-2-3



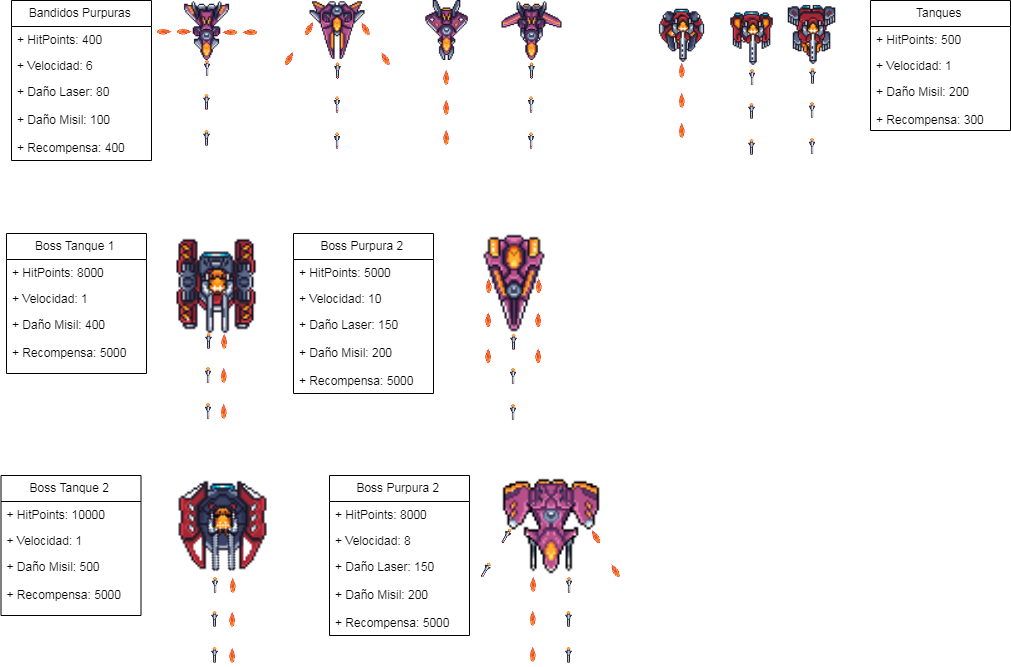
Stages 2-3-4



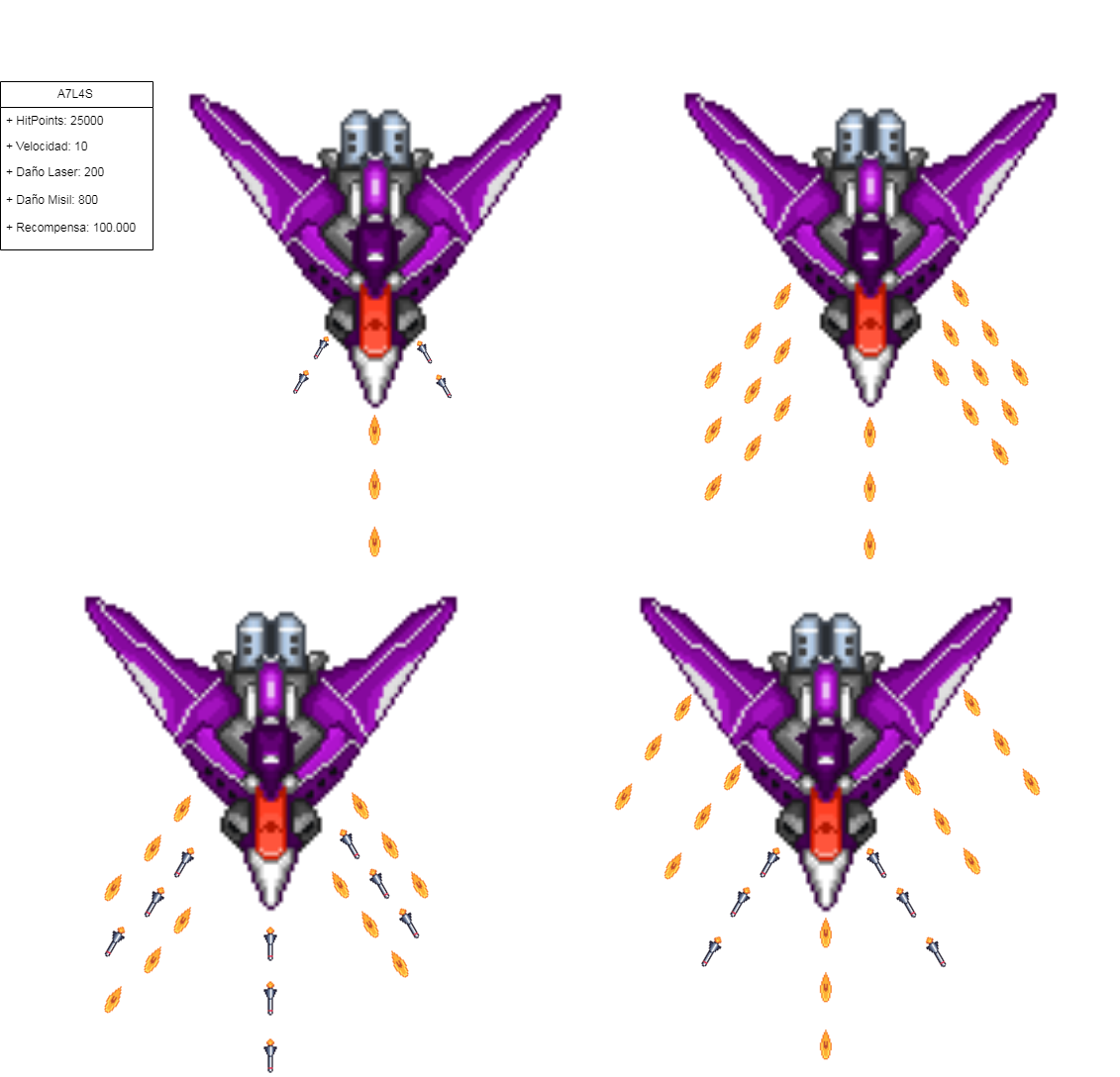
Stages 3-4



Stage 4 y final



Stage Final



Los bosses parciales y finales de cada stage solo pertenecen a su propio stage, mientras que los bandidos pueden repetirse en distintos stages (como se detalla).

Los bandidos saldrán de la siguiente manera: estarán incorporados en un script de “oleada” el cual contiene el tipo de enemigo y el camino que va a seguir ese conjunto. Entonces “oleada” . El script que controla todo esto es el de respawn, el cual tiene una lista de oleadas, con lo cual tendremos distintas respawn de enemigos distintos con distintos caminos. A continuación adjuntamos el código que hace posible esto:

Codigo principal que maneja el rewspawn

public class EnemySpawner : MonoBehaviour

{

[SerializeField] private List<WaveConfig> waveConfigs;

[SerializeField] private int startingWave = 0;

private IEnumerator Start() {

do {

yield return StartCoroutine(SpawnAllWaves());

} while (looping);

}

private IEnumerator SpawnAllWaves() {

for (int waveIndex = startingWave; waveIndex < waveConfigs.Count; waveIndex++) {

var currentWave = waveConfigs[waveIndex];

yield return StartCoroutine(SpawnAllEnemiesInWave(currentWave));

}

}

private IEnumerator SpawnAllEnemiesInWave(WaveConfig waveConfig) {

for (int enemyCount = 0; enemyCount < waveConfig.GetNumberOfEnemys(); enemyCount++) {

var newEnemy = Instantiate(waveConfig.GetEnemyPreFab(),

waveConfig.GetPathPrefab()[0].transform.position,

Quaternion.identity);

newEnemy.GetComponent<EnemyPath>().SetWayConfig(waveConfig);

yield return new WaitForSeconds(waveConfig.GetTimeBetweenSpawns());

}

Codigo que crea la oleada

public class WaveConfig : ScriptableObject {

[SerializeField] private GameObject enemyPreFab;

[SerializeField] private GameObject pathPreFab;

[SerializeField] private float timeBetweenSpawns = 0.5f;

[SerializeField] private float spawnRandomFactor = 0.3f;

[SerializeField] private int numberOfEnemys = 5;

[SerializeField] private float moveSpeed = 2f;

public GameObject GetEnemyPreFab() {

return this.enemyPreFab;

}

public List<Transform> GetPathPrefab() {

var waveWayPoints = new List<Transform>();

foreach (Transform wayPoint in pathPreFab.transform) {

waveWayPoints.Add(wayPoint);

}

return waveWayPoints;

}

public float GetTimeBetweenSpawns() {

return this.timeBetweenSpawns;

}

public float GetSpawnRandomFactor() {

return this.spawnRandomFactor;

}

public int GetNumberOfEnemys() {

return this.numberOfEnemys;

}

public float GetMoveSpeed() {

return this.moveSpeed;

}

Codigo que itera sobre el camino de los enemigos

public class EnemyPath : MonoBehaviour

{

private List<Transform> wayPoints;

private WaveConfig waveConfig;

private int wayPointIndex = 0;

public void SetWayConfig(WaveConfig waveConfig) {

this.waveConfig = waveConfig;

}

private void Start() {

wayPoints = waveConfig.GetPathPrefab();

this.transform.position = wayPoints[wayPointIndex].transform.position;

}

private void Update() {

Move();

}

private void Move() {

if (wayPointIndex <= wayPoints.Count - 1) {

var targetPosition = wayPoints[wayPointIndex].transform.position;

var moveThisFrame = waveConfig.GetMoveSpeed() \* Time.deltaTime;

this.transform.position = Vector2.MoveTowards(this.transform.position, targetPosition, moveThisFrame);

if (this.transform.position == targetPosition) {

wayPointIndex++;

}

}

else {

Destroy(gameObject);

}

# Árbol de habilidades

El árbol de habilidades de progresión de la asimov propuesto es el siguiente:



Los puntajes propuestos para alcanzar cada nivel salen de la siguiente tabla donde se intenta darle un aproximado matemático a la cantidad de enemigos que se necesita por nivel para lograr el tiempo deseado de juego, teniendo en cuenta que aproximadamente cada enemigo (sin incluir bosses) debería tardar entre 5 y 8 segundos para destruirlo.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stage | Boss | Puntos | Tiempo deseado (sin boss) | Enemigos destruidos | Puntos hipotéticos |
| 1.1 | Rojo1 | 1.000 | 0:10:00 | 75 | 10.000 |
| 1.2 | Rojo2 | 1.000 | 0:10:00 | 75 | 11.500 |
| 1.3 | BL00D | 2.500 | 0:15:00 | 115 | 19.750 |
| 2.1 | Verde1 | 1.500 | 0:15:00 | 115 | 22.200 |
| 2.2 | Verde2 | 1.500 | 0:20:00 | 150 | 31.500 |
| 2.3 | H0RN3T | 3.500 | 0:20:00 | 150 | 36.500 |
| 3.1 | Gris1 | 4.000 | 0:25:00 | 190 | 61.000 |
| 3.2 | Gris2 | 4.000 | 0:25:00 | 190 | 64.800 |
| 3.3 | SH4K3 & B4K3 | 6.000 | 0:30:00 | 225 | 80.250 |
| 4.1 | Tanque1 | 5.000 | 0:30:00 | 225 | 86.000 |
| 4.2 | Purpura1 | 5.000 | 0:35:00 | 270 | 107.600 |
| 4.3 | Tanque2 | 5.000 | 0:35:00 | 270 | 107.600 |
| 4.4 | Purpura2 | 5.000 | 0:40:00 | 300 | 125.000 |
| 4.5 | A7L4S | 100.000 |  |  | 763.700 |

Ese total de puntos supondría un run sin muertes (que quitan el 15% de puntos) ni regeneraciones (cuestan 10% de puntos) y destruyendo todas las naves enemigas.

Se tuvo en cuenta que para completar el árbol de niveles se precisan los siguientes puntos:

Nivel 1 🡪 25.000 puntos

Nivel 2 🡪 40.000 puntos (65.000 acumulados)

Nivel 3 🡪 75.000 puntos (140.000 acumulados)

Nivel 4 🡪 130.000 puntos (270.000 acumulados)

Nivel 5 🡪 200.000 puntos (470.000 acumulados)

Nivel 6 🡪400.000 puntos (870.000 acumulados)

De esto se desprende que muy difícilmente en un primer run se puedan obtener todas las habilidades del árbol y muy probablemente esto suceda luego de finalizar el juego (con los 100.000 puntos que da el jefe final).

# Condiciones de Victoria/Derrota

La condición de victoria del juego consiste en despejar los 4 stages que tiene el juego y para lograrlo debe destruir a los semi-jefes como a los jefes de stages. Incluyendo el boss final del juego.

Las condiciones de derrota es que el jugador desee dejar de jugar ya que no existe la permadeath, el sistema de muertes funciona con los puntos, desarrollando:  
El jugador tiene 2 parámetros fundamentales: 3 vidas de (al inicio) 250 hitpoints en modo ofensivo o 500 en modo defensivo cada una y los puntos que va sumando por destruir enemigos (por ejemplo un enemigo inicial suma 100 puntos). Los puntos pueden ser utilizados de 3 formas, 2 son voluntarias y una involuntaria: de manera voluntaria puede utilizarlos para comprar niveles del árbol de habilidad o para regenerar una vida (cuando tenga menos de 3) lo cual cuesta un 10% de los puntos al momento de la regeneración. De manera involuntaria se produce al ser destruido, por cada vida de las 3 que pierda el jugador se le quita el 2,5% de puntos, y al llegar a la 3ra se lo penalizar volviendo al inicio del stage parcial (por ejemplo el 1.3) y con un 12,5% más de puntos (lo que totalizan 15% al perder la 3ra vida). Si la cantidad de puntos es 0 no se descuenta.

Con lo cual el jugador puede reiniciar un stage las veces que sea necesario pero la pérdida de puntos sucesiva hará más y más difícil el avance.

Con este balance se busca alentar al jugador a no acumular puntos y utilizarlos en compra de niveles, así como a llevarlo a decidir si prefiere gastar un 10% para regenerarse o arriesgarse a perder una vida por menos (2,5%) a riesgo de perderlas todas y tener que reiniciar el stage y perder 15%.

En cuanto a las subidas de nivel, estas solo pueden hacerse al finalizar cada semi-stage (bosses parciales o finales de stage) y no en cualquier momento dado. Con esto se busca amplificar el estilo estratégico del manejo del puntaje.

En el punto definitivo lo que perdura del juego es el puntaje, es decir a dos usuarios que terminaron la partida quien fue mejor o peor lo determinan sus puntos restantes, teniendo en cuenta que el Boss final da 100.000 puntos este es el puntaje mínimo que un vencedor puede tener. Con esto cerramos el argumento definitivo de que usar puntos es una cuestión estratégica y no simplemente una moneda de intercambio. Muchas variables (muertes, niveles, regeneración, enemigos destruidos) van a afectar el puntaje final del jugador.

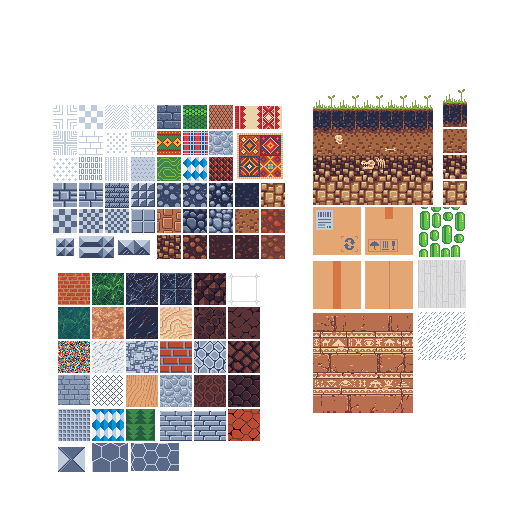
# Mundo

El mundo del juego se divide en 4 stages bien definidos, el estilo grafico es (al igual que los sprites) 8 bits – pixel art.

Las siguientes imágenes son solo representativas y no las definitivas para el juego, antes que nada se adjunta un spritesheet con el estilo deseado (y factible de ser utilizado).

Como el juego es del estilo vertical scroll la vista es top down, pero los escenarios no van a ser con una visión cenital (totalmente desde arriba) sino con una leve perspectiva inclinada para darle un poco mas de cuerpo al ambiente.

Estilo:





Stage 1: La Selva

Es el nivel introductorio del juego, aquí los colores tienen que ser vivos y fuertes, relajantes y lo mas contrastantes con el ultimo nivel (que se detallara al final). Como los enemigos son de color celestes metálico, naranja y rojo no hay problemas de contraste.



Stage 2: Montañas

Las montañas son el siguiente nivel después del bosque, aquí los colores son un poco mas opacos sin llegar a ser oscuro, los enemigos son celestes, naranja, rojos y verdes con lo cual sigue sin haber inconveniente de constrastes.



Stage 3: Ruinas

Las ruinas son el nivel que sucede a las montañas y tienen que ser un poco más oscuros sin dejar de lado un verde opaco (como restos de vegetación) y con estructuras viejas derruidas. Los enemigos son rojos, verdes y grises. El verde de los enemigos es un verde de tono manzana mientras que la (escasa) vegetación debe ser de un tono militar con lo cual hay una diferencia significativa.



Stage 4: Ciudadela

La ciudadela es la última zona del juego y la de mayor contraste con respecto a la primera, los colores tienen que ser opacos, oscuros y metalizados y transmitir una sensación de tensión visual. Tiene que haber grandes estructuras pero sin signos de vida humana. Los enemigos son verde, gris y purpura. Hay que tener precaución con los enemigos grises para que no se confundan con el escenario, si bien dichos enemigos tienen muchos vivos rojos que ayudan a la identificación visual



# Historia

## Breve propuesta:

El contexto histórico del juego se centra en un futuro no muy lejano. La supuestamente desaparecida ciudad de la Atlántida en realidad era una isla híper tecnológica con ocultamiento visual. Esta ciudad estaba habitada por humanos bastamente más inteligentes que el resto de personas en el planeta tierra, cualquier habitante común de la Atlántida podía ser considerado un genio para una persona común del mundo exterior. Los avances tecnológicos de la Atlántida habían sido ampliamente utilizados por distintas potencias mundiales (que sabían de su existencia y comerciaban con ellos pero nunca divulgaron el conocimiento), estos avances eran en su mayoría del tipo armamentista y de robótica. La insipiente oleada de robots con inteligencia artificial que empezaba a inundar el planeta tierra (para realizar tareas mundanas) tenían su origen allí sin que nadie lo supiera. La opera prima de los atlantes era la A7L4S, una inteligencia artificial capaz de controlar prácticamente todos los aspectos de necesidades que se pudieran satisfacer a través de las matemáticas, ya desde su primera iteración (.1) prácticamente todo estaba controlado por ella en la ciudad. Iteración tras iteración “Atlas” (como era conocida de manera interna) fue adquiriendo mayores capacidades, y en su fatídica iteración 13 tras conectarla a la red mundial de internet cobro conciencia propia desatando un ataque nuclear a puntos estratégicos del planeta, utilizando a las IA que podía controlar para hacerse cargo de la ciudad de la Atlántida (y así protegerse físicamente) y erigiéndose como una especie de protectora de la humanidad.

La historia de nuestro personaje (DG) empieza muchos años después, el era un niño muy pequeño cuando todo esto sucedió, se crio en un mundo ya devastado y donde se sobrevive más de lo que se vive. Se quedo sin padres a una muy corta edad y junto con su hermana fueron criados por su tío. Este fue quien le enseño a pilotear, disparar y el oficio del comerciante. Su talento natural combinado con el maestro superlativo que tuvo lo llevo a posicionarse como uno de los comerciantes más requeridos en todos lados.

El inicio del viaje esta dado por una oferta irresistible a nivel económico de llevar un cargamento misterioso (que por contrato no se puede abrir antes de la entrega) pero será rechazada por el lugar de destino: la Atlántida. Hay pocos que se animen a entrar allí por temor, en el caso de DG es por el profundo rechazo que le produce, ya que culpa del estado del mundo a la inteligencia artificial Atlas.

Posteriormente a rechazar la carga descubre que quien si acepto la tarea fue su hermana, la cual ya debería haber regresado, con lo cual se apresura a buscarla.

Una vez en el lugar rescatada desde los escombros de la nave destruida a su hermana, la cual le explica la situación, el cargamento contenía un material metálico recientemente descubierto que es virtualmente indestructible. ¿Qué necesidad tendría una inteligencia artificial para hacerse de dicho material? DG no lo sabe, pero nadie se mete con su familia y vive para contarlo, ni siquiera una maquina que no considera “viva”.

## Horas de gameplay

La consideración general es que el juego dura, en un run casi perfecto, unas 5 horas aproximadamente.

Un jugador nuevo con todos los procesos de prueba/error debería estar en 7-8 horas.

# Interfaz de Usuario

La interfaz de usuario debe ser lo más minimalista posible ya que la pantalla se va a encontrar constantemente muy cargada de sprites y muchos de ellos se moverán a gran velocidad, aun mas importante al ser un juego de implica mucha concentración y reflejos es contraproducente cargar la pantalla con UI ya que puede distraer o molestar a la visual del usuario.  
Como propuesta la idea es mostrar los siguientes elementos fundamentales en la parte superior de la pantalla (los tamaños son solo representativos así como los colores pero hay una escala aproximada entre unos y otros):

Vidas restantes, hitpoints restantes, Puntos, Stage.

2/3 25.000 135 1.3

life.png

# Musica

El estilo de la música para ser consonante con el estilo grafico va a ser del tipo retro 8 bits. Tiene que pasar lo mas desapercibido posible (para no desconcentrar al jugador) pero al mismo tiempo mantener la tonalidad de velocidad/riesgo constante, ayudar a la tensión y diversión del juego.

Es importante respetar este estilo sin resignar calidad, con todas las herramientas que existen es posible lograr el espíritu retro pero con muchas mejoras cuantitativas y cualitativas respecto a antiguas épocas mucho más limitadas tecnológicamente.

Los efectos de sonidos (disparos, impactos, etc.) son muy importante porque son una asistencia para el jugador, al ser un juego donde los reflejos y la velocidad juegan un papel importante el jugador no puede interpretar visualmente todas las cosas que suceden en la pantalla al mismo tiempo, por eso es muy importante que mediante asistencia con el audio para (sin tener que mirar) comprender que destruyo una nave enemiga, o que está siendo disparado desde un sector u otro, así como si fue impactado.