Between Lights

Por:

Erik Locutura
Pablo Martinez
Rodrigo Ben
José Miguel Arranz
Gemma Bachs

Índice:

Título:

Between Lights

Plataforma:

PC

Género + Nicho:

Plataformas 2D + Puzzle + Exploración

Cliente y Público Objetivo:

Nuestro cliente objetivo son jugadores casuales que busquen un juego diferente, a su vez familiares que busquen un regalo alejado de una estética violenta, para regalar a un hijo o nieto, o sencillamente quieran regalarse a sí mismos un buen rato.

En cuanto al usuario, nos enfocamos en jugadores casuales y también jugadores que conozcan bien el género de puzzle plataforma y quieran experimentar algo diferente a lo que existe actualmente en el mercado.

(El PEGI es 7).

Fecha de entrega:

Se dispone de 10 semanas para realizar el proyecto. De las cuales una se dedicará a la redacción del GDD, aparte de reservar una segunda semana para imprevistos.

Previsto su lanzamiento al mercado para Navidad a fin de aprovechar la oleada de consumismo que caracteriza estas fechas.

Modelo de negocio:

Premium. El precio se estima en 9,99 euros, disponible en Steam.

Competencia:

3 brillantes:

- *Hollow Knight (Team Cherry, 2017)*: bella estética gótica y bella paleta de colores además de buenas mecánicas.
- *Limbo (Playdead, 2010)*: estética oscura, particular, pronto reconocible, historia intrigante, mecánicas interesantes.
- *Grim Fandango (LucasArts, 1998)*: música muy acertada, ambientación en el mundo de los muertos (estética Halloweenesca).

y 3 nefastos:

- *Bubsy in Fractured Furry Tales (Imagitec Design, 1994)*: control torpe, plataformas torpes. Cada entrega iba de mal en peor.
- *Ninjabread Man (Data Design Interactive, 2005):* controles y gráficos confusos, historia muy corta, fácilmente completable en media hora.
- *Page Chronica (Red Hare Studios, 2012):* mala mezcla de plataformas con puzzles, difícil de enganchar el jugador.

Cámara:

Cámara estática simple, el que se mueve es el protagonista.

Historia:

La historia transcurre en una noche de los años 50. Encarnas a Jorgito, un niño cuyo padre acaba de sufrir un ataque al corazón. Pese a haber sido reanimado, no responde a nada, sus ganas de vivir se han perdido por los cielos. A partir de una historia contada por tu anciana abuela, irás en busca de una antigua máscara, reliquia familiar, y descubrirás que, al ponértela, puedes viajar al Más Allá, y así ir en busca de los trozos del alma pérdida de tu padre, con la finalidad de retornarla al Mundo De Los Vivos y que él viva.

Para completar tu cometido contarás con un tiempo límite (el reloj avanza, ¡no dejes que amanezca sin tener a tu padre de vuelta!), y máscaras de 4 tipos distintos, que irás encontrando en el Mundo De Los Vivos. Cada una de ellas te otorgará la habilidad de pasar al Más Allá y usar un determinado poder que necesitará de un recurso. Éste recurso (energía) lo obtendrás de los altares en el Mundo De Los Vivos.

Controles y principales mecánicas:

Control por teclado.

- A+D: desplazamiento
- Espacio: saltar
- E: interactuar con objetos del entorno y NPCs.

Habilidades.

- Click izquierdo: activación de la principal habilidad de la máscara en uso.
- Click derecho: activación de la habilidad secundaria de la máscara en uso.

Modo de juego:

Singleplayer offline aventura.

Mascaras:

- 1a máscara:
    - Habilidad principal: altera el estado físico del personaje durante un tiempo limitado de manera que las corrientes de aire puedan empujarle/propulsarle.
    - Habilidad secundaria: permite al personaje atravesar ciertos tipos de estructuras durante un tiempo limitado.
- 2a máscara:
    - Habilidad principal: permite al personaje interactuar con NPCs para descubrir pistas y/o descubrir zonas nuevas.
- 3a máscara:
    - Habilidad principal: permite al personaje ver cosas ocultas a simple vista, revelando partes de la zona actual o zonas ocultas.
    - Habilidad secundaria: permite al personaje crear estructuras en puntos determinados para superar obstáculos.
- 4a máscara: es la combinación de las anteriores y te abrirá las puertas al último nivel.

Niveles:

Cada mundo es un nivel, con lo cual, en total habrá 4 niveles más el mundo puente que es desde donde se empieza la aventura:

1. Omio

Paraíso del sol, lugar del gozo permanente. La gente canta y baila hasta que a los 4 años se va volando en forma de ave donde vuelven a reencarnar. Se divisan colinas y un firmamento cristalino y bello. El mar reluce tanto que parece de oro.

Colores que lo representan: amarillo, verde, oro.

2. Taloc

Paraíso de la lluvia, lugar de reposo y abundancia. Aquí van todos aquellos que mueren por causas relacionadas con el agua. Numerosos árboles frutales pueblan los campos de verde hierba. Los habitantes siempre van mojados, a causa de las continuas lluvias; dicha circunstancia no les representa una incomodidad.

Colores que lo representan: azules, plata.

3. Auco

Mundo infantil donde reposan los bebés. Encontrarás un árbol gigante que llora leche. En teoría inocente y claro, pero a medida que uno se va adentrando en el

lugar, se descubren inesperadas y atroces habitudes; por ejemplo, la extraña afición que tienen esos bebés a vomitar la leche ingerida para fabricar figurillas a modo de juguete.

Colores que lo representan: pastel, blanco.

4. Mitlan

Mundo muy oscuro, cerrado y sin ventanas. Acaban aquí los muertos por causa natural; no sufren, tan sólo se aburren a muerte. Yacen apoyados en las negras paredes de piedra, sin mucho que decir y nada que hacer, y prácticamente no se relacionan. Acostumbran a ser viejos hastiados.

Colores que lo representan: negro, gris, marrón.

Programación:

Lenguaje C# para Unity.

Scripts:

Interactuables

- Interactable

  Interficie que asegurara que todos los interactuables tendrán la función Interact(PlayerController).

- Altar

  Tendrá 2 variables integer, una para el número de máscara al recargar y otra para la cantidad de cargas que puede recargar.

  La función Interact(PlayerController) usará la función RechargeMask(int) int de la clase Mask para recargar la máscara especificada en la variable dentro del PlayerController y restar lo retornador a las cargas restantes.

- HiddenInteractable

  Tendrá una variable bool revealed y una lista de GameObjects desactivados.

  La función Reveal() asignará este bool a true y activará el GameObject del sistema de partículas para mostrar que se ha revelado.

  La función Interact(PlayerController) comprobará que la variable revealed sea true, en caso afirmativo, llamar a la función SetActive(true) de todos los GameObjects en la lista y se cambiará la variable revealed a false, también desactivando el sistema de partículas. En caso que la variable revealed sea false, mostrará una mensaje de error al interactuar.

- Collectable

  La función Interact(PlayerController) llamara a la función GotNewCollectable() de la clase LevelController actual.

- StructureSpawner

  Tendrá una variable GameObject toSpawn que será el objeto a hacer aparecer y una variable int para la cantidad de segundos que permanecerá la estructura en el escenario.

  La función Interact(PlayerController) llamara la función SetActive(true) del GameObject toSpawn y iniciara una coroutine para llamar SetActive(false) después de los segundos especificados.

- DialogNPC

  Tendrá una lista con los diálogos a mostrar y un int index del diálogo actual mostrado.

  La función Interact (PlayerController) llamará la función ShowNewDialog(string) de la clase UIController y sumará el index del diálogo actual. Al llegar al último index de la lista, se llamará la función HideDialog() de la clase UIController y establecerá el index a zero.

  La función OnTriggerExit() llamara la función hideDialog() de la clase UIController.

Controllers

- InteractableSystem

  Mantendrá una lista de GameObjects en rango de interacción con la funciones OnTriggerEnter/Exit().

  La función InteractWithClosest(PlayerController) llamara a la función Interact(PlayerController) de el Interactable de la lista que esté más cerca del personaje.

- FollowObjectController

  Tendrá una variable GameObject y cinco variables int, tres para offsets de los ejes y las dos restantes para un máximo y un mínimo que puede desplazarse en el eje Y.

  En LateUpdate() usará las variables anteriores para seguir la posición del GameObject sumando los offsets, restringiendo la posición del eje Y de la cámara al máximo y mínimo establecidos.

- GamePostProcessingController

  Se encargará de controlar las animaciones del post processing

- MasksController

Tendrá una variable para cada máscara, así como una lista de booleans para determinar qué máscaras han sido obtenidas y se pueden usar, y una variable para controlar qué máscara se está usando actualmente.

En Update () se encargará de controlar el tiempo de equipado de las máscaras así como de recolectar el input para la activación de estas.

Al recibir un input para una máscara, comprobará si ya tiene una máscara activa, en este caso, se la quitara y pasará a usar la "máscara" NoMask, en caso de no tener ninguna activa, se equipara la seleccionada.

- LevelController

  Tendrá dos variables GameObject que serán los contenedores de los objetos de cada mundo.

  Tendrá dos variables para el tiempo de inicio y el de fin del nivel actual, así como una referencia al TimeSystem y una variable con el tiempo actual de la partida.

  Tendrá un contador de collectables para controlar el progreso del jugador.

  Se llamará a la función GetTime() con un invokeRepeating cada segundo, la cual hará un GetTime() GameTime de la clase TimeSystem y, en caso de ser superior a la variable de fin del nivel actual, se perderá la partida, enviando un triggerEvent de fin de partida.

  La función SwapWorld() activará y desactivará el contenedor de gameobjects correspondiente en función de si cambia al mundo de los vivos o el de los muertos.

  La función PauseGame(bool) llamara la función TimeStopped(bool) para para el tiempo de la partida y también bloqueará/desbloqueará todo tipo de input del PlayerController, MaskController y InteractSystem.

  La función GotNewCollectable() sumará al contador de collectables obtenidos.

- PlayerController

  Tendrá las variables de speed y jumpSpeed.

  En Update() se encargará de analizar el input de movimiento y trasladarlo a la clase base PhysicsObject mediante un override de la función ComputeVelocity().

- TimeSystem

  Tendrá una variable uint para controlar el tiempo en segundos de la partida, y una variable int que se usará de multiplicador del paso del tiempo.

  Tendra las funciones GetTime() GameTime, SetTime(GameTime) y AddTime(GameTime). GetTime() al formato GameTime de dd:hh:mm:ss. SetTime y AddTime usarán el GameTime proporcionado para fijar o sumar esa cantidad de segundos al tiempo actual.

- UIController

  Controlara los GameObjects contenedores de los diferentes componentes de la UI, con funciones para mostrar y ocultar dichos GameObjects.

Máscaras

- Mask

  Interficie que se asegurara que todas las mascaras tienen las funciones Ability(), SecondaryAbility(), EndOngoingEffects() y RechargeMask(int) int.

- MaskAbstract

  Clase base para las máscaras, con las variables totalUses y remainingUses.

  Desarrollara la función RechargeMask(int) int ya que tiene que funcionar igual para todas las máscaras, esta deberá calcular las cargas que faltan hasta “llenar” la máscara, en caso de que esta diferencia sea mayor o igual al numero cargas proporcionadas, sumar las cargas proporcionadas a remainingUses y devolver este mismo número, en caso de que la diferencia sea menor a el número de cargas proporcionadas, sumar la diferencia a remainingUses y devolver esta diferencia. Ambos casos activaran el efecto de partículas rechargeParticleSystem del PlayerController.

  La función EndOnGoingEffects() se encargará de establecer todos los booleans de habilidades activas a false, específico para cada máscara.

- NoMask

  Default que se asignará cuando no se esté usando ninguna máscara.

- Mask1

  Tendrá dos variables float para las duraciones de los efectos de las habilidades y dos variables boolean para marcar si estos están activos. Además de una variable extra para cada efecto de partículas de cada habilidad.

Las funciones Ability() y SecondaryAbility(), al ser llamadas, en caso de que la máscara tenga usos restantes, se le restará uno y se cambiará el estado del boolean pertinente a true, activará el sistema de partículas pertinente, e instanciará una coroutine para que, después del tiempo especificado en la variable, cambie ese mismo boolean a false y desactivando el respectivo sistema de partículas para finalizar el uso de la habilidad.

- Mask2

  Tendrá una variable ParticleSystem para el efecto de partículas de la habilidad de esta máscara.

  La función Ability(), en caso de que la máscara tenga usos restantes, se le restará uno y se llamara a la funcion RevealDialog() de todos los NPCs actualmente en pantalla, así como llamar a la función Play() del sistema de partículas una sola vez.

- Mask3

  Tendrá una variable float para la duración del efecto de la primera habilidad y un boolean para la activación de la segunda habilidad.

  La funciona Ability(), en caso de que la máscara tenga usos restantes, se le restará uno y se llamará a la función Reveal() de todos los objetos de la clase HiddenInteractable en pantalla.

  La función SecondaryAbility() , en caso de que la máscara tenga usos restantes, se le restará uno y se cambiara el estado del boolean a true, dejandote interactuar con Interactables del tipo StructureSpawner.

- MaskFinal

  Tanto la función Ability() como la SecondaryAbility() cargaran la escena final del juego.

Varios

- PhysicsObject

  Se encargará de aplicar gravedad y movimiento al rigidbody del GameObject base mediante la funcion ComputeVelocity().También comprobará si el objeto está grounded, así como colisiones mediante el rigidbody y un array de RayCastHits..

- GameTime

  Clase que contendrá uints para días, horas, minutos y segundos.

  Tambien tendra la funcion HigherThan(GameTime) bool.

- PassThroughObject

En la función OnCollisionEnter(Collision) comprobará que el colisionador en cuestión tenga un MaskController, se esté usando la máscara mask1 y si se está usando actualmente la habilidad secundaria mediante su boolean, en ese caso, la variable IsTrigger del collider del PassThroughObject se le asignará true, dejando a cualquier objeto traspasarlo.

En la función OnTriggerExit(Collider) comprobará que el Collider que esté dejando de estar en contacto tenga un MaskController, en caso afirmativo, la variable IsTrigger del collider de este objeto pasara a ser false.

- Wind

  Tendrá la variable Vector2 windSpeed para controlar la dirección y fuerza del viento.

  En la función OnTriggerStay(Collider) se comprobará que el colisionador contenga PhysicsObject y MaskController, en caso afirmativo se comprobará que se esté usando la mask1 y variable abilityActive de esta sea true, dadas estas condiciones verdaderas, se le añadirá la velocidad del viento Vector2 a la velocidad del PhysicsObject colisionador.

- ZoneSwitcher

  Tendrá una variable Transform que se usará para saber la destinación del objeto.

  La función OnTriggerEnter(Collider) comprobará que el collider en cuestión sea el jugador, en ese caso, establece su posición a la de la variable transform.

NPCs sin inteligencia artificial.

No habrá dificultad adaptativa.

Gems / Unique Selling Points:

Viajar entre los diferentes planos del Más Allá, basados en el folclore mexicano y la cosmogonía mesoamericana.

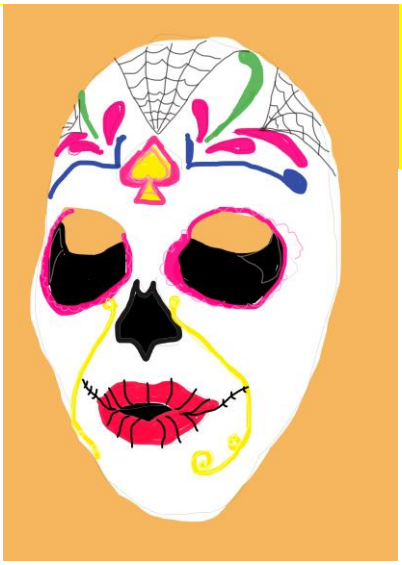
Combina mecánicas entre el Mundo De Los Vivos y su representación en el Más Allá para resolver puzzles.

Personajes, concepts, mockup:

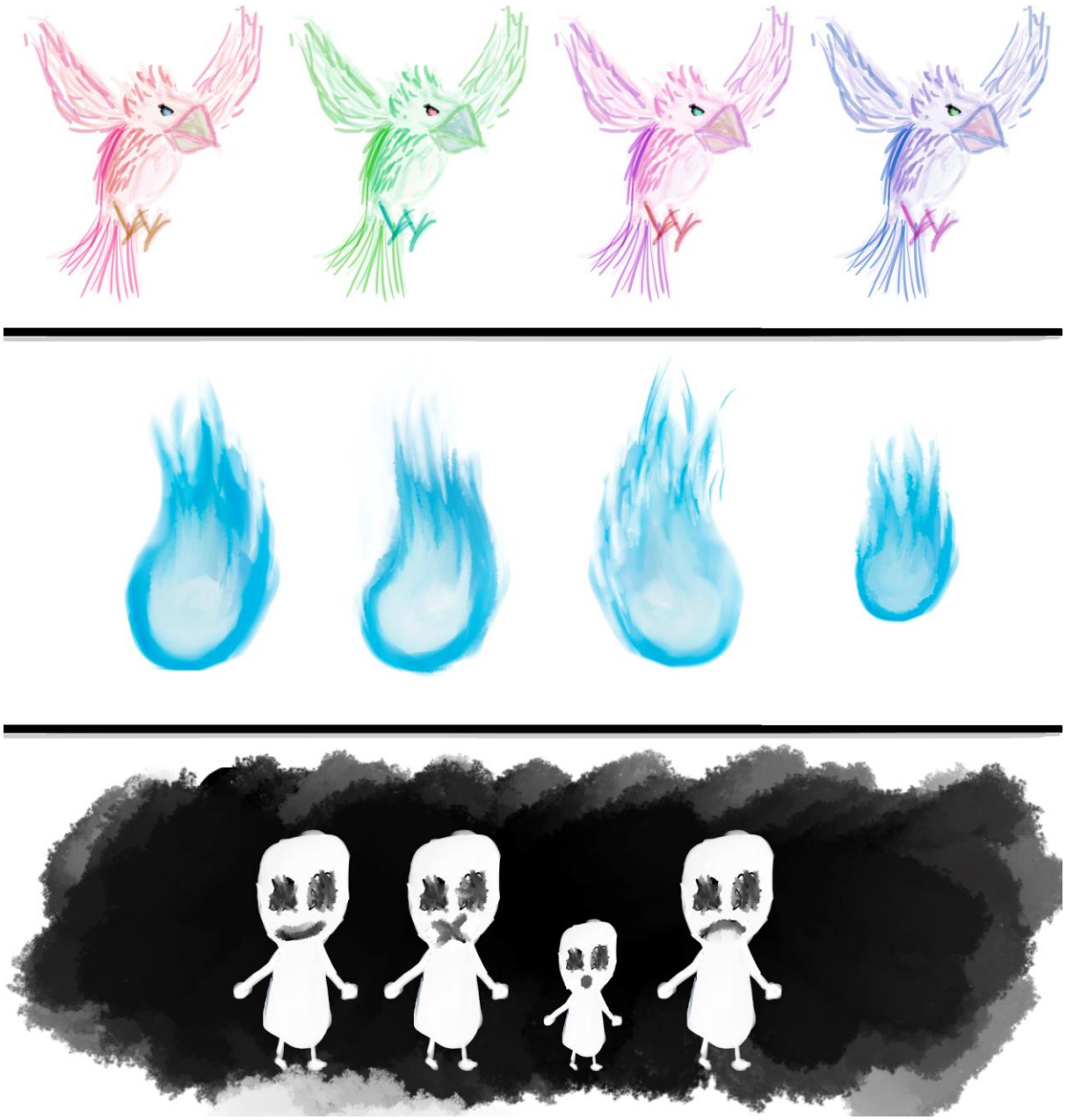
- Jorgito: el protagonista. Empieza un viaje para recuperar la vida de su padre.
- Padre de Jorgito (neutral): se encuentra entre la vida y la muerte a causa de un repentino ataque al corazón.
    - Abuela de Jorgito (ayudadora): le cuenta a Jorgito una historia sobre una máscara, reliquia familiar.
    - Muertos (ayudadores): Jorgito habla con ellos para conseguir pistas y/o desbloquear partes nuevas del nivel.

Mundo De Los Vivos

Omio / Taloc / Auco / Mitlan



Máscaras



Espíritus de los diferentes mundos

UI (Interfaz de Usuario)

Level Design (Nivel 1)

Toda la parte artística presentada son *Art Concepts* de elaboración propia.