Videojuego basado en el sonido.

Justificación.

Agradecimientos.

Dedicatoria.

Resumen.

Diseño del juego.

Desarrollo del juego.

Diagrama de clases.

Sonido:

* Sonido 2D.
* Sonido Visual

Herramientas:

* Visual Studio.
* Box2D.
* SFML.
* Tiled.
* Gimp.
* Audacity.
* TinyXML.

Mecanicas:

* Sonidos como mecánica.
* Caminar.
* Sigilo.
* Mecanismos y puertas.
* Rocas.

Fisicas:

* Gravedad 0.
* Mascaras de colisiones.

Motor Visual:

* VertexArray
* TriangleStrips

Patrones:

* RAII.
* NonCopyable.
* Façade.
* Fabric.

Lenguaje:

* C++11

Videojuego basado en el sonido.

Justificación.

Agradecimientos.

Dedicatoria.

Resumen.

Echoes es un juego de terror que utiliza una formula sencilla: eliminar la visión y agudizar el oído. Consiste en recorrer una serie de mapas formados por salas totalmente a oscuras, para ello tendremos que guiarnos únicamente por el sonido de nuestros pasos, buscando la salida sin morir en el intento. Suena fácil, pero estaremos rodeados de peligros y cada mapa será más complicado que el anterior.

Diseño del juego.

Solamente seremos capaces de situar visualmente nuestros pies, que serán unas huellas blancas en la pantalla totalmente negra, esto será así mientras no nos movamos, porque el sonido de nuestros pasos, o cualquier otro sonido, generará una serie de ondas visuales que nos permitirán vislumbrar la forma del mapa y los peligros que nos rodean, rebotando en las paredes y cambiando de color dependiendo de la zona por la que pasen.

Estas ondas sonoras serán más intensas y duraderas cuanto más fuerte sea el sonido producido, y a lo largo de los mapas se irán añadiendo mecánicas para generar más sonido, o menos, según nos convenga, lo que da una profundidad y posibilidad de creación de puzles infinita.

Desarrollo del juego.

Diagrama de clases.

Sonido:

* Sonido 2D.
* Sonido Visual

Herramientas:

* Visual Studio.

Para la realización de este proyecto se ha optado por utilizar el entorno de desarrollo Visual Studio 2015 puesto que después de haber probado diferentes opciones a lo largo de la carrera, como son Codeblocks o eclipse, lo considero muy superior en calidad de las herramientas de depuración y ayuda a la hora de escribir código mas rápidamente.

* SFML.

Como motor gráfico se ha usado SFML por considerarlo una herramienta excelente que aporta tanto herramientas como tutoriales y ayudas para usarlas, es estupendo a la hora de utilizar gráficos en 2D y aporta librerías de audio, eventos y control de tiempo.

* Box2D.

Puesto que es un juego que requería de una componente física muy potente, con manejo de mascaras de colisiones, sistema de partículas y resolución de colisiones complejas y rápidas se ha apostado por la librería de físicas Box2D, a la que se le ha dotado de una fachada con el patrón façade para poder sustituir la librería de manera rápida en caso de necesitarlo.

* Tiled.

Para la realización de los diferentes mapas del juego se ha utilizado la herramienta tiled, de la cual se han utilizado únicamente las capas de objetos, esta herramienta te permite dibujar en una rejilla la forma del mapa deseado utilizando capas para distinguir entre los diferentes objetos y guarda la información en un fichero de texto XML, el cual leeremos de la manera que nos convenga desde la aplicación y podremos construir nuestros mapas.

* TinyXML.

Como ya se ha comentado, al utilizar la herramienta Tiled para crear los mapas se requiere de una librería de lectura de ficheros XML, y para ello se ha utilizado tinyXML, que permite recorrer un fichero en este formato a través de las etiquetas de manera fácil e intuitiva.

* Gimp.

Para la realización del sprite de los pies del personaje se ha utilizado la herramienta Gimp, que es un potente entorno de diseño grafico gratuito.

* Audacity.

Con Audacity se han modificado y ajustado los audios capturados para el juego para adaptarlos a las necesidades que teníamos al realizar un juego de terror, como ecos o reverberaciones.

* GitHub y SourceTree

Como repositorio y sistema de control de versiones se ha utilizado una cuenta en GitHub y el entorno grafico SourceTree de Atlassian.

Mecanicas:

* Sonidos como mecánica.

El sonido será la principal fuente de información de lo que sucede en el mundo que nos rodea, este será representado con líneas rectas cuya velocidad, intensidad y duración dependerán del la fuerza y volumen del sonido que la produce. Esto dará como resultado una pantalla cargada de información a través de estas ondas que rebotando contra los límites del mapa nos darán información de hacia donde tenemos que ir y que nos rodea, ya que cuando una de estas ondas pase sobre los diferentes objetos del juego (palancas, pinchos, etc) se colorearan durante el periodo en el que están situadas sobre el objeto de un color característico del objeto, por ejemplo, la salida de cada mapa se coloreara de verde.

* Caminar.

El simple hecho de caminar producirá ruido a causa de nuestros pasos, estos generaran una buena cantidad de ondas sonoras que nos harán de guía en nuestro camino.

* Pinchos

Tendremos que tener cuidado de hacia dónde nos dirigimos, ya que la mazmorra estará llena de pinchos (rojo) que tendremos que sortear.

* Enemigos.

A lo largo y ancho de los mapas nos iremos encontrando con unos peculiares enemigos, de los cuales no conoceremos su aspecto puesto que no los vemos, pero si les escuchamos, y ellos a nosotros, ¡tened cuidado!

* Sigilo.

A veces caminar es demasiado peligroso si nos acechan enemigos, para esto tenemos la capacidad de avanzar sigilosamente y sin hacer casi ruido, lo malo es eso mismo, ¡que sin ruido no vemos hacia donde estamos yendo!.

* Mecanismos y puertas.

A lo largo de nuestro camino nos encontraremos con una serie de mecanismos (color amarillo) que al pisarlos nos darán acceso a nuevas zonas antes inaccesibles.

* Rocas.

Cuando una zona está vigilada por un terrible enemigo tendremos la posibilidad de lanzar una piedra para distraerle, puesto que siempre van a olfatear el lugar del cual les llega un ruido.

* Agua.

El agua es un enemigo “silencioso”, no nos matará pero nos hará avanzar más despacio y hacer más ruido, tened mucho cuidado con ella.

Fisicas:

* Gravedad 0.

Aunque se utilice un motor de físicas para el proyecto no necesitamos gravedad, puesto que es vista de pájaro y no veremos los objetos subir ni bajar, solo rebotar contra las paredes.

* Mascaras de colisiones.

Dada la complejidad de las colisiones y la gran variedad de objetos con diferentes respuestas a las colisiones dependiendo que tipo de objeto choca con cada uno de ellos, se ha implementado un sistema de mascaras de colisiones para indicar a box2d que colisiones son sensores, cuales producen una colisión real y que resultado tiene cada una de las combinaciones.

Motor Visual:

* VertexArray
* TriangleStrips

Patrones:

* RAII.
* NonCopyable.
* Façade.
* Fabric.

Lenguaje:

* C++11

Metodologia: