**Do Cakes also fall from the Sky? (Homenaje a Moving Out)**

Sergio David Trujillo Becerra 1202355

Richard Andrés Arandia Beltrán 1202207

David Josué Pinto Gómez 1202363

Universidad Militar Nueva Granada

Facultad de Ingeniería, MULB (Simulación)

Docente Gabriel Ávila

Bogotá, D.C. 1 de septiembre 2023

**Resumen**

En esta investigación, se expone a modo homenaje, una representación del videojuego “Moving Out”, donde se realiza un análisis de sus físicas implementadas, como movimientos de aceleración/velocidad, parabólico, masa, gravedad, caída libre, colisiones, fuerzas, partículas, entre otras, para tomarlas como base en el desarrollo de un videojuego propio con nueva temática de jugabilidad. Para esto, se implementa el desarrollo en el motor de videojuegos multiplataforma de Unity.

**Homenaje a Moving Out en sus Físicas Implementadas**

Los videojuegos se han convertido en un medio de entretenimiento masivo para todo tipo de edades, particularmente, los simuladores de entorno de la vida real, que toman como base fundamental en su desarrollo, la física. En una amplia gama de videojuegos basados en la física, Moving Out no es la excepción, ya que refleja movimientos físicos de tiro parabólico, fuerza de fricción, leyes de Newton, simulación de partículas (implementación de atributos físicos), entre otras características de la física inmersas en la mudanza de objetos y su interacción con los personajes.

En este proyecto, se postula una propuesta de un videojuego a desarrollar, tras el análisis de las físicas identificadas en el videojuego Moving Out, para posteriormente implementarlas en una realización propia simulada en el motor de videojuegos multiplataforma de Unity.

A través del estudio y el desarrollo, se pretende poner en práctica los conocimientos de la física-teórica al ser llevados en la máxima expresión inmersiva de los videojuegos simulados en Unity.

**Objetivos**

**General**

Desarrollar un videojuego implementando sistemas físicos inspirados en el videojuego de Moving Out.

**Específicos**

* Extraer las físicas del videojuego Moving Out creado, para crear un videojuego propio.
* Analizar las ecuaciones correspondientes de cada modelo físico implicado en el videojuego, con el fin de implementarlas en una simulación.
* Poner en práctica los conocimientos teóricos físicos en la asignatura de simulación.

**Marco Teórico**

**¿Qué es Moving Out?**

Es un videojuego donde el usuario asume el rol de un empleado de una empresa de mudanza, donde tendrá que mover los objetos de la casa del cliente a su camión de mudanza y viceversa. Este juego se caracteriza por su jugabilidad, su arte y diseño en cuanto a la ambientación y sencillez de sus gráficos bastante cómicos (Moving Out, 2021).

**¿Qué Físicas tiene?**

Se analizan las siguientes físicas:

* Leyes de Newton (inercia, dinámica, acción-reacción), en la fuerza de empuje y estado de objetos.
* Fuerza de fricción y rozamiento contra el suelo.
* Colisión de objetos al momento de moverlos y chocarlos con otros.
* Dinámica de vidrios quebrados cuando se pasa un objeto a través de estos.
* Movimiento parabólico, al momento de arrojar las cajas en un punto en específico. Como también identificar el rectilíneo, en el desplazamiento de cajas-objetos.
* Caída libre, en algunos objetos del entorno, cuando este se cae o da paso al jugador.
* No tiene demasiado en cuenta la gravedad con respecto al peso de todos los objetos.
* Velocidad, aceleración; dirección.
* Partículas al momento de interactuar con otros objetos (confeti de un buzón) o cuando estos se están movimiento (humo que sale de sus pies).
* Se tiene en cuenta la masa, aunque es la misma en todos los objetos.

**¿Qué Videojuegos Similares hay?**

* Overcooked 1 y 2: Al momento de interactuar con los objetos en la cocina.
* Tools up: Similitud en partículas, fricción, tipo parabólico.
* Unrailed!: Al momento de mover los objetos e ir construyendo las vías del tren y cuando el tren de descarrila.
* Human fall Flat: Interacción con los objetos y personajes, como si todo fuera lo mismo o bien, tuviese el mismo peso.

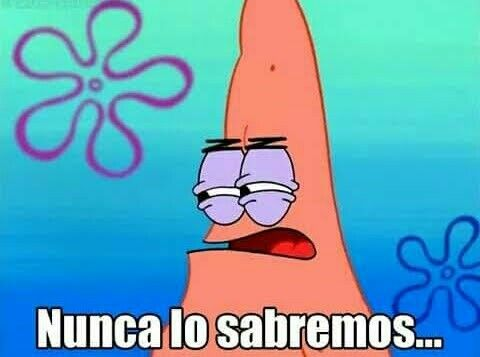
***Propuesta del Videojuego a Desarrollar (Primera versión)***

Do Cakes also fall from the Sky?:En una pastelería, hay dos personas trabajando: el que cocina y el que reparte. El que reparte es quien atiende y da los pedidos a los clientes, éste tiene que ir atendiendo a cada uno para llevarlos a la cocina donde su compañero/a le haga el pedido, claramente se tardará x tiempo en sacar el o los pasteles. Deberá ir limpiando la zona por el desastre que dejan los clientes y seguir atendiendo a los demás. A veces tendrá que limpiar, atender y llevar pasteles a la vez para entregarlos rápido a sus clientes mientras que cada vez hay menos tiempo para todo: más clientes, más velocidad, pero...más pasteles acumulados, más basura que limpiar y si no limpia, se resbala, y algunos pasteles se caen, etc.

Se implementará tomando los modelos físicos estudiados para aplicarlo en Unity.

***Propuesta del Videojuego a Desarrollar (Versión final)***

En una pastelería, una maquina saca cada cierto tiempo un pastel, el repartidor acepta un trabajo que consiste en ir a la zona asignada del lugar y debe dejar el pedido allí mismo para acumular puntos; pero si el repartidor del lugar no es tan rápido, no se acumularán los puntos sino los pasteles, y cuando se acumulan, estos pierden su equilibrio debido a lo alto que esta la pila de los pasteles hasta que al final ya no hay más pasteles para repartir (a no que los tome del suelo y siga con su trabajo). El repartidor deberá ir atravesando obstáculos y ser paciente mientras llega su plataforma móvil en todo ese gran lugar donde la máquina trabaja para entregar los pasteles. ¿Los entregará a tiempo? ¿Se caerá la pila de pasteles? ¿El repartidor estará soñando y pensará que los pasteles caen del cielo? ¿Por qué los pasteles salen de una máquina del cielo?



**Etapas del desarrollo.**

**Lluvia de ideas.**

Trabajamos para tener las ideas más simples, pero a la vez no tan complejas de hacer, estas ideas se centran en varios aspectos que se pueden visualizar en el videojuego Moving Out: Las colisiones con objetos y el jugador, las ventanas, agarre de objetos y entrada y salida a través de puertas, partículas de salto (humo de impulso).

En nuestro caso, es recrear lo anterior, pero agregando algunos cambios: Los objetos agarrar será un o varios pasteles.

**Lista de ideas posibles.**

Al final, para nuestro pequeño juego decidimos hacer las puertas, ventanas, plataformas móviles, interacción con pasteles (para nuestro juego).

**Repartición de trabajo.**

Entonces tenemos la siguiente lista de lo que hay que agregar para el juego.

* Generación de pasteles.
* Interacción con objetos sólidos.
* Generación aleatoria de colores en los pasteles.
* Interacción con puertas mediante las físicas de Unity como lo es: RigidBody.
* Puerta de carnicero.
* Jugador con gravedad.
* Movimiento del jugador.
* Salto del jugador.
* Cámara del jugador.
* Interacción con el vidrio.
* Plataformas móviles.
* Partículas de choque o de salto.

**Avances.**

Los avances que teníamos que entregar por semana están listados por las tareas que tenía que hacer cada integrante del grupo.

**Parte de Richard.**

* Generación de pasteles.
* Generación aleatoria de colores en los pasteles.
* Solución de balanceo y falta de equilibrio en los pasteles.
* Sistema de puntuación de los pasteles.
* Modelos de sillas, cajas y mesa en un intento cartoon.

**Parte de Sergio.**

* Interacción con la ventana.
* Físicas de la ventana.
* Físicas de la puerta de carnicero.
* Físicas y colisión del jugador con las puertas normales.
* Partículas de salto al jugador.

**Parte de David.**

* Movimiento del jugador.
* Primer mapa de prueba.
* Objetos sólidos con los que interactúa o “tropieza” el jugador (sillas, mesa, cajas).
* Interfaz/UI del menú de pausa.
* Interfaz/UI del menú principal.
* Idea del mapa final y como se distribuyen los objetos a lo largo del mapa.

**Tiempo de entrega.**

Respuesta corta: Avance semanal.

Respuesta larga: Cada semana mostrábamos avances de la parte de cada uno y terminábamos haciendo una retroalimentación al final de la clase aportando ideas o soluciones de algunos inconvenientes, ya sea buscando al profesor y preguntando algunas pequeñas cosas, o bien, ideando algunas soluciones para los scripts.

**Mapa final.**

Decidímos hacer un mapa un poco más grande que el de pruebas que habíamos hecho en un principio. Con esto, ya solo nos faltaría por ajustar los objetos y prefabs que ya están configurados en el Unity (la idea es solo moverlos o cambiarlos de posición).

**Retoques finales.**

Como lo dice el meme:” A veces cuando planeas una cosa, te sale otra completamente diferente.”



Este sabio refrán hace referencia a que teníamos que luchar para solucionar algunos detalles de último minuto para ajustar esas pequeñas “bobadas” que nos habían surgido ya que, al combinar los scripts, unir los objetos o renombrarlos mal a lo largo del proyecto, pues nos ha llevado un dolor de cabeza completar nuestro producto final.

O simplemente errores al exportar mal los avances para adecuarlos a los proyectos que teníamos repartidos en nuestro grupo.

**Juego final.**

A parte de que quedó bonito y algo bien hecho con mucho cariño y esfuerzo, se tiene un producto donde se aplicaron varios conceptos donde:

* Sí, si se aplican las colisiones vistas en clase y optimizadas por Unity.
* Sí, se tuvo en cuenta las interacciones con los objetos del juego Moving Out.
* Sí, aprendimos que no es tan fácil hacer un juego bien pulido en un semestre.

**Diseño, implementación de ideas y problemas con soluciones del juego.**

**Parte de Richard.**

1. **Modelos iniciales de objetos en Autodesk Maya:**

Se modelaron 3 objetos, correspondientes a una silla, una mesa y una caja, los cuales serían los obstáculos que debería sortear el jugador.

Cuando se importaron los modelos a unity, se les agregó un componente de rigidbody con una masa correspondiente al objeto con el fin de alterar el movimiento del jugador y un box collider para que se hiciera la respectiva colisión.

1. **Script del seguimiento de la cámara al jugador:**

Se realizó un script el cual inicializaba una posicion (x,y,z) de la cámara la cual siempre apuntaría a la posición del jugador con respecto al tiempo.

Se realizó un zoom al momento de estar la cámara en reposo por unos segundos y se quitaba el zoom cuando el jugador volviera a moverse.

1. **Generación de pasteles:**

Inicialmente se creó un script que clonara el objeto correspondiente al pastel, y se daba un número máximo de iteraciones, se pensó hacerlo de forma aleatoria, sin embargo, finalmente se decidió ir generando los pasteles, uno sobre otro de forma que se hiciera una torre de pasteles y gracias a sus componentes de rigidbody empezaran a perder equilibrio y balancearse de un lado a otro a medida que iba creciendo la misma torre.

Sin embargo, el principal inconveniente fue que, al generarse un pastel sobre la posición del último pastel generado para hacer la torre, cuando el jugador agarraba el ultimo pastel generado, el siguiente pastel se generaba sobre el que tenía el jugador, debido a que la posición en x,y,z del último pastel generado cambiaba con el tiempo al ser llevado por el jugador.

Para esto se pensaron dos soluciones, los pasteles debían generarse siempre en una sola posición, sin importar si el jugador tenía agarrado un pastel o no, y la segunda solución era elaborar un condicional cuya función seria comprobar si el jugador agarro el ultimo pastel generado, y si era así entonces basarse en la posición del suelo del pastel generado anteriormente al último de la torre.

Finalmente se utilizó la primera solución y se dio un color aleatorio a cada material de los pasteles generados.

1. **Incorporación del movimiento de oscilación de los pasteles al apilarse en la mano del jugador.**

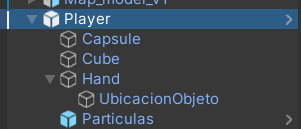
Inicialmente se quería crear un efecto de perdida de equilibrio a medida que la torre de los pasteles iba siendo mayor, para esto se pensó en tomar la torre de pasteles como una pila, y darle a esta un movimiento senoidal en 3D, sin embargo, fue un código bastante complejo de elaborar, debido a que los pasteles no oscilaban como se esperaba.

Debido a esto, se optó por realizar una analogía más sencilla donde después de que el jugador agarrase el 3er pastel, lo que hiciera el script fuera empezar a realizar movimientos muy sutiles de un lado a otro desde el primer objeto apilado, hasta el último objeto apilado, cuyos movimientos fueran más notorios, según la altura en la que estaba el objeto(pastel).

**Parte de David.**

1. **Creación del jugador.**

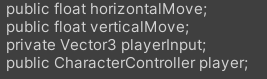
Para el jugador se tuvo en cuenta lo siguiente: Una vista casi isométrica donde se le permita recorrer y tropezar entre otros objetos. Vale, en ese caso se programa los movimientos en un script encargado de realizar las interacciones con el player controller y su respectivo rigidbody. Pero, no es solo una píldora sin más, está conformado por más cosas: Un grupo padre llamado jugador, la famosa píldora, un cubo que hace referencia a la vista del jugador, la mano la cual esta es transparente y se utiliza para poner los pasteles como arreglo heredado. Las partículas se mencionan más adelante.



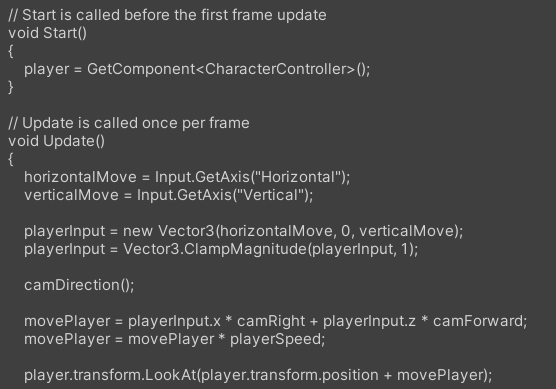
1. **Movimiento del jugador**

Se crea una variable llamada playerSpeed, el cual es un número que será multiplicado por el DeltaTimepara luego obtener los ejes X y Z y permitir mover al jugador por el área de la zona del mapa. Así mismo, se crean otras variables encargadas de utilizar el CharacterController de player (el jugador), las variables horizontalMove y verticalMove, pero solo manejamos horizontalMove por el momento.



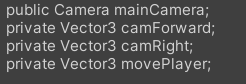


Luego de eso, se realiza la transformación de la posición del jugador.



1. **La cámara del jugador.**

Esta cámara es especial, pues esta sigue al jugador por donde vaya y como vaya. Para eso, se toma la cámara y se renombra para adicionarle el movimiento de la cámara con respecto al jugador.



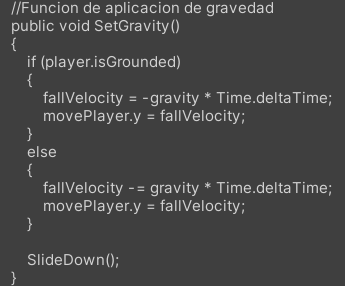
Luego se realiza algunas transformaciones para la cámara mediante una función llamada camDirection() sin alterar el eje y, ya que este es controlado a futuro por el salto del jugador.

1. **La gravedad del jugador.**

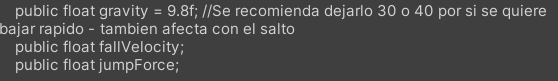
Ahora, nuestro jugador debe saltar, para eso se le asigna una variable que le da el valor a la gravedad para que sea multiplicada por el movimiento y del jugador, una fuerza de salto y su función como una habilidad que tiene el jugador.

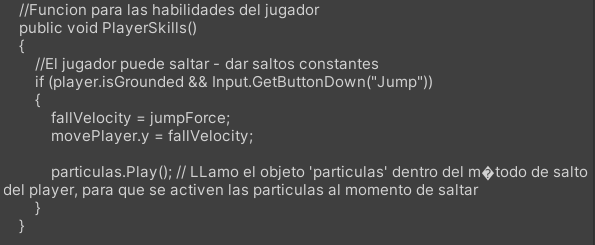


Pero antes, se debe aplicar la gravedad para el jugador, esto se realiza mediante una función detectando si el jugador está en una plataforma, o no para más adelante aplicarle otra variable encargada de dar, la velocidad de caída.



Y así mismo, se hace el proceso contrario, para realizarle el salto con otra función nominada PlayerSkills, la cual se le da a la velocidad de caída, la fuerza del salto y al movimiento Y del jugador, fallVelocity nuevamente.

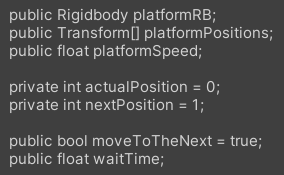




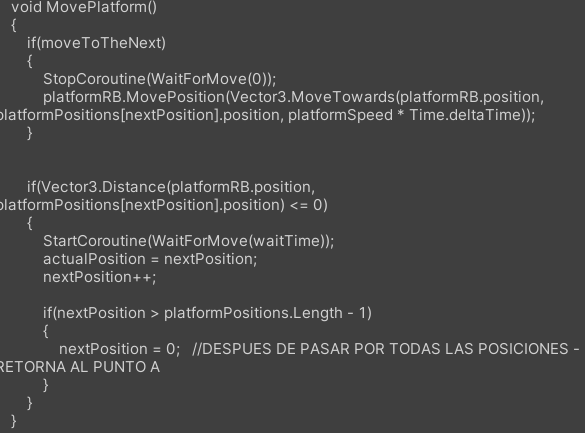
1. **Las plataformas para el jugador.**

Para las plataformas, simplemente se creó 2 scripts adicionales: moveWithFloors para el jugador, y PlatformController para la plataforma. Empecemos con el segundo, ya que es lo principal (No hay movimiento en plataforma si no hay plataforma).

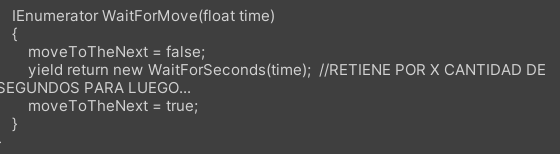
En PlatformController se crearon variables que le asigna velocidad de movimiento, arreglo con puntos o paradas, el rigidbody de la plataforma, la primera/temporal/última posición, tiempo de espera.



Con eso se mueve la plataforma con una transformación de un vector de un lado al otro calculando su respectiva distancia para que esta se mueva hacia esa dirección.

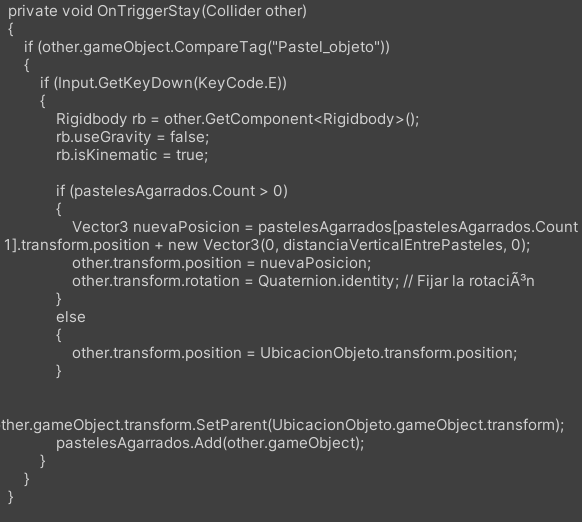
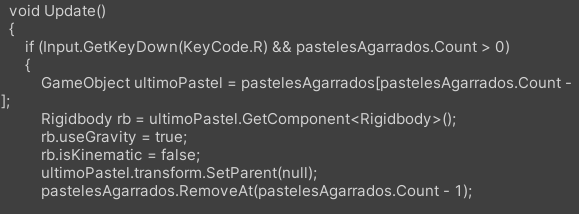


Al final se le asigno el tiempo de espera, solo hace un ciclo indicando que posición sigue y cuál es la final, para que llegue al primer punto, se detenga, continue con el siguiente, así sucesivamente.



1. **Los pasteles del jugador.**

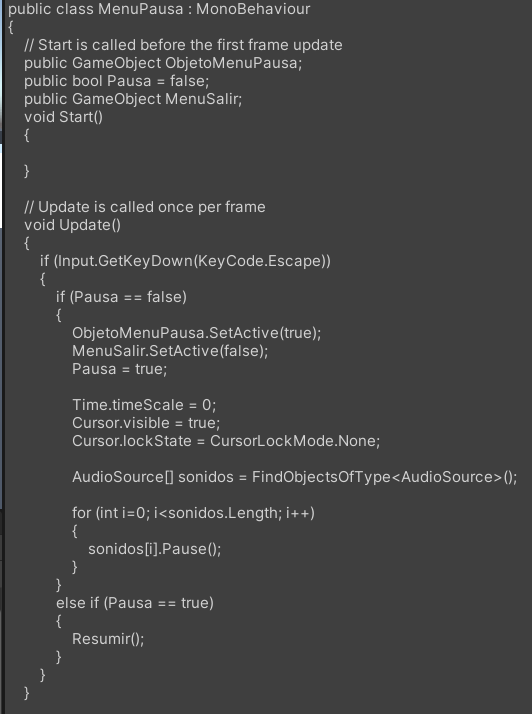
Aparte de la oscilación, se realizó principalmente el agarre del jugador con los pasteles. Esto se logra con las variables que es un arreglo que detecta estos pasteles, toma uno y luego hace caer el resto de los que se han generado en el sitio. Se le cambia las propiedades para que el pastel sea estático con y durante el jugador para que cuando lo deje en otro lado, este regrese a tener las características padres de su lugar inicial, pero sin intervenir con el arreglo principal de donde se cuenta y se hace la pila de los pasteles.

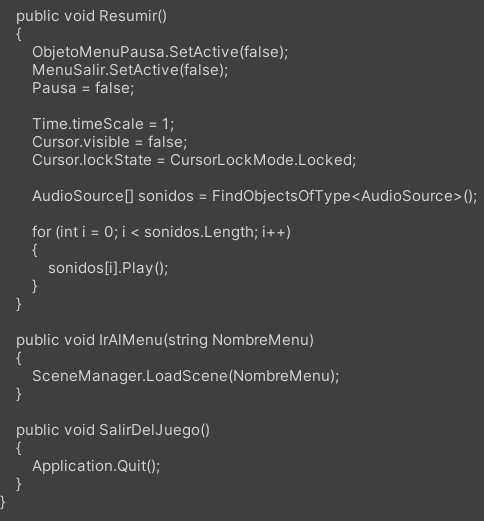


1. **Las UI del juego.**

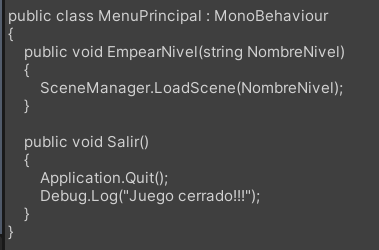
Para las UI solamente se utilizó el gameObject de la UI ya disponible de Unity, se acomodó, se ordenó y se creó su respectivo script que interactúe con el esc de pausa, los clics para salir o redundar el juego. En el menú de inicio lo mismo, y que este redirija del menú al nivel de prueba. Así mismo, en la pausa se detuvo tanto juego en su DeltaTime.







*UI y script del menú de pausa.*





*UI y script del menú principal.*

**Parte de Sergio.**

1. **Partículas del Jugador**

Las partículas del jugador fueron creadas añadiéndole un objeto con componente de particle system, se adecuaron propiedades necesarias como renderer de sprites, dirección, rotación, velocidad inicio, tiempo de vida, color, especificando que sean generadas en la misma posición en la que están pero no sigan al jugador (simulation space), etc. Anexándolas al jugador, se crea el script para el funcionamiento de generación automática con condicionales "tan pronto salte (se oprima la tecla de espacio en teclado)".

1. **Puertas**

Para la puerta normal (sola) y la puerta doble, se crean objetos bisagra para los objetos de rectángulos creados, añadiéndoles componentes box collider, rigidbody sin gravedad. A los objetos-empty de bisagra se añade hinge joint, vinculando objetos de los rectángulos y, modificando propiedades de límite de apertura, ángulo rotación, spring para efecto de retorno a la posición original, boucinnes y demás.

Para la puerta de cintas, se crean objetos en cadena de rectángulos en columnas, con uno base-padre. Añadiéndoles box collider y hinge joint, con un ángulo limitante de rotación de mayor a menor (es decir, que los rectángulos en columna superiores tienen 30°, hasta los últimos rectángulos de 90°, para su efecto poscolisión de apertura o desplazamiento de las cintas).

1. **Ventana**

Se hace previo modelado en Maya, por partes: marco de la ventana, ventana entera y ventana fragmentada (polígonos de vidrios individuales). Se exporta a Unity y se añaden sus componentes; rigidbody, box collider. Para la ventana fragmentada, se añade a cada vidrio/polígono individual, mesh collider y su trigger activado. El vidrio entero contiene el script; en él se tuvo que especificar en un condicional que al detectar alguna colisión (objetos externos que incidan a colisionar), se instancie la ventana fragmentada en su misma posición, mediante 'quaternion.identity', y que destruya/elimine el vidrio entero; así se creó el efecto visual de rompimiento de la ventana.

Claramente se tuvo dificultad visual en la pos-colisión de la ventana, pues al momento de colisionar, los vidrios correctamente caen, pero se perciben unas columnas poligonales realmente extrañas...se revisó el modelado en Maya, la exportación, las propiedades en los componentes añadidos en los prefab u objetos de cada parte de la ventana (más aún, reanalizando la ventana fragmentada, sus polígonos/vidrios individuales y mesh).

1. **Qué y cómo se han realizado las pruebas.**

**Prueba de pasteles.**

Para los pasteles se tuvo en cuenta lo siguiente, y al final terminó ocurriendo:

* Generación de pasteles cada cierto tiempo
* Colores al azar en los pasteles
* Colisiones del jugador-pastel
* Interacción del jugador con el pastel.
* Salida del pastel del arreglo como hijo del jugador.
* Caída del pastel y retorno de las características cuando el jugador lo deja en otro lado del mapa.
* Acumulación de los pasteles.
* Los pasteles no se sobreponen entre sí, sino que se apilan.
* Los pasteles se caen si uno de estos los empuja.
* Los pasteles tienen interacción entre ellos.
* Los pasteles al aumentar de tamaño su pila, este pierde el equilibrio (se luchó por conseguirlo profe).
* La pila de pasteles si se cae (se luchó por conseguirlo profe).

**Prueba de jugador.**

Para el jugador se tuvo en cuenta lo siguiente, y al final terminó ocurriendo:

* Script del movimiento del jugador.
* Script de la gravedad del mundo y del jugador.
* Salto del jugador.
* El jugador se podría mover en las plataformas móviles.
* El jugador, debido a los cambios con los scripts de los pasteles, no se mueve por la plataforma, pero no la atraviesa ni se bugea con ella.
* El jugador ya no colisiona y hace saltos raros con los objetos o pasteles.
* El jugador puede atravesar las puertas.
* Las puertas eran pesadas para el jugador, tenía que hacer mucha fuerza o durar mucho tiempo entre ellas para atravesarlas.
* El jugador puede colisionar y mover de un lado a otro con los objetos sólidos.
* Si el jugador interactúa con la mesa, este le costará más; pero si interactúa con la caja, le costará menos.
* El jugador puede agarrar y soltar el o los pasteles.
* El jugador tiene el pastel como una clase heredada por la clase del vector de la generación de los pasteles.
* Las características de los pasteles generados no intervienen con los del jugador.

**Prueba de puertas.**

Para las puertas se tuvo en cuenta lo siguiente, y al final terminó ocurriendo:

* No se tuvo en cuenta ningún script.
* Se utilizaron modelos 3d del mismo Unity, rectángulos y cubos.
* Se crearon las bisagras para que sean los hinge-joint de los rectángulos de 1 tipo de puerta (la normal), la sencilla puerta normal de toda la vida.
* Con los componentes, se alternaron las propiedades de rotación, límite de apertura (ángulo de rotación) para hacer la puerta doble del tipo 2.
* Un spring para que la puerta tipo 2, la puerta doble normal. “rebote” o que retorne a su posición inicial.
* La puerta tipo 3, la del carnicero (la de cintas) se tuvo en cuenta los ángulos de rotación para que la cadena de mini rectángulos se movieran acorde a la colisión del jugador.
* Cada columna de cadena de mini rectángulos del pureta tipo 3, están anexadas a la base principal la cual es el rectángulo grande por encima de estas (es como el padre y las cintas las hijas).

**Prueba de vidrio.**

Para el vidrio se tuvo en cuenta lo siguiente, y al final terminó ocurriendo:

* Se hizo un modelo en Autodesk Maya, el marco de la ventana, la ventana entera y sus fragmentos.
* Una vez se tienen los fragmentos como prefabs en Unity, se organizaron uno al otro, tanto el marco de la ventana, la ventana entera y la ventana fragmentada.
* Se procede a tomar la ventana fragmentada para anexarle un mesh, para adaptarlo a la forma de cada polígono individual.
* Se realiza el script para la ventana entera el cual consiste donde se crea una condición para que al momento en que detecte una colisión de un objeto externo (como el jugador o los pasteles), esta se “destruya”.
* Luego de eso, aparece los fragmentos de la ventana “destruida”, casi como una transición de un modelo a otro, pero con esos fragmentos listos para interactuar con el entorno.

**Conclusión**

El análisis de la mecánica física del videojuego Moving Out, permite extrapolar el conocimiento de la física teórica en la implementación de un videojuego propio simulado.

**Referencias**

*Moving Out*. (28 de diciembre de 2021). Epic Games Store; Epic Games. https://store.epicgames.com/es-ES/p/moving-out

*Overcooked | Cooking Video Game | Team17*. (16 de junio de 2023). Team17 Digital LTD - the Spirit of Independent Games. Recuperado el 25 de agosto de 2023 de https://www.team17.com/games/overcooked/

‌*Tools Up!* (2022). Nintendo.com. https://www.nintendo.com/es-co/store/products/tools-up-switch/

*Unrailed! - A co-op Multiplayer Railroad Construction Game*. (2021). Unrailed-Game.com. Recuperado el 25 de agosto de 2023 de https://www.unrailed-game.com/