

Reporte de Documentación

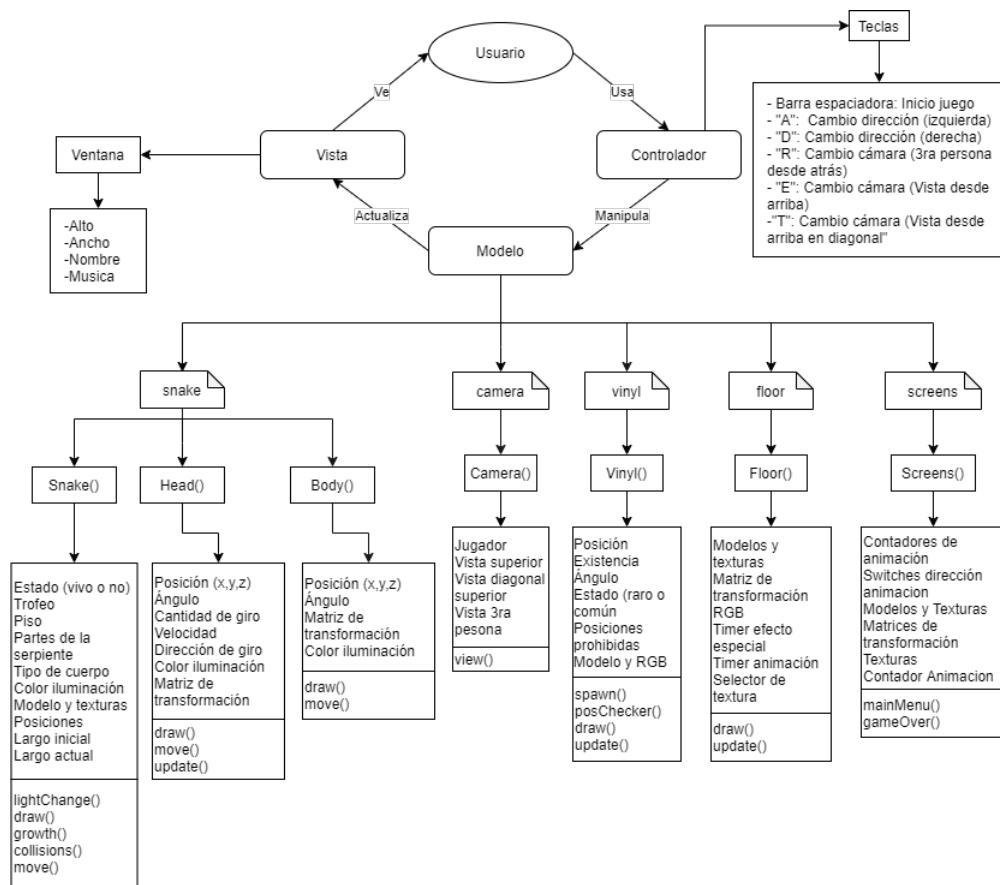
Gonzalo Alarcón

17 de diciembre de 2020

1. Solución Propuesta

1.1. Arquitectura de la solución

Se utilizó el patrón de diseño Modelo-Vista-Controlador , donde el usuario usa el controlador para manipular el modelo y este actualiza la vista que finalmente vera el usuario que puede nuevamente usar el controlador creandose asi un bucle.



1.1.1. Modelo

El Modelo se basa en cinco archivos distintos: `snake.py`, `camera.py`, `vinyl.py`, `floor.py` y `screens.py`; y cada uno de estos contiene una o más clases. A continuación se explicaran brevemente en que consisten las clases de cada archivo.

`snake.py`

- **Head()**: Esta clase corresponde a la “cabeza de la serpiente”, y se inicializa con `self.x`, `self.y`, `self.z`, que son las coordenadas ($x = 0.0, y = 0.0, z = -4.5$), en las que se encuentra la “cabeza”; el ángulo (`self.theta`) de rotación en el que mira la cabeza de la serpiente respecto al eje que va de la cabeza a los pies de la esta; la cantidad de giro (`self.bend`), que corresponde a que tan “brusco” gira la “cabeza de la serpiente”, a mayor cantidad de giro más cerradas seran las vueltas que de la “serpiente”; la velocidad de la serpiente (`self.front`), que corresponde a la velocidad con que se mueve hacia donde está mirando (y a su vez el “donde está mirando” depende de `self.theta`); la dirección de giro (`self.turn`), derecha o izquierda, hacia la cual estará girando la “serpiente”; el color de iluminación (`self.weirdLight`) que corresponderá al color con el que será iluminada la cabeza de la serpiente; y la matriz de transformacion (`self.transform`) que corresponderá a las transformaciones en el espacio que se le aplicaran a la “cabeza”.

Además esta clase cuenta con varios métodos: `draw()` dibuja la “cabeza”; `move()` desplaza la “cabeza” y finalmente `update()` actualiza el ángulo en que mira la “cabeza”.

- **Body()**: Esta clase corresponde a cada parte del “cuerpo de la serpiente”, que para nosotros será cada parte que no sea la cabeza, se inicializa con elementos muy similares a `Head()`, `self.x`, `self.y`, `self.z` dictan la posicion de la parte del “cuerpo”; `self.theta` indica el ángulo en mira; `self.transform` es la matriz de transformación y `self.weirdLight` es el color de iluminación.

Además esta clase cuenta con métodos similares a `Head()`: `draw()` dibuja la “parte del cuerpo” y `move()` actualiza su posicion.

- **Snake()**: Esta clase se inicializa con un bool (`self.alive`) que indica si la serpiente esta viva ((`self.alive = True`)) o muerta (`self.alive = False`); el objetivo (`self.objective`) que debe recolectar la serpiente , que en este caso será el objeto vinilo que se creará; el piso (`self.floor`) en el que se moverá la serpiente, que en este caso será la pista de baile/cima del edificio; una lista (`self.snake_parts`) de las partes que componen a la serpiente; una lista (`self.bodyTypeList`) que indica que textura le corresponde a cada parte de la serpiente; el color de la iluminación de la escena (`self.weirdLight`) para que el color de la serpiente coincida con el de la escena; el modelo (`obj`) y las texturas 5 (`bodyOBJ1-bodyOBJ5`); los dibujos 5 (`self.GPU1 - self.GPU5`) de las partes de la serpiente, un deque (`self.positions`) que corresponde a las posiciones por las que ha

pasado la serpiente; el largo inicial (`self.initial_size`) de la serpiente; y el largo (`self.length`) actual de la serpiente, ademas se agregan objetos (`Body()`) a la lista `self.snake_parts` hasta alcanzar el `self.initial_size` y se desplazan para que las partes queden separadas a cierta distancia unas de otras.

Esta clase cuenta con varios metodos `lightChange()` cambia el color con el que se ilumina la serpiente, `draw()` invoca el metodo `draw()` de cada parte de cuerpo dependiendo que valor le esta asociado en la lista `self.bodyTypeList`, `growth()` crea una parte nueva, la agrega a `self.snake_parts` cambia su posicion, invoca su metodo `move()`, le asigna una textura aleatoria de las 5 disponibles agregando un numero aleatorio entre (0,4) a `self.bodyTypeList` y finalmente le suma 1 a `self.length`; `collision()` revisa las colisiones de la serpiente con su cuerpo, con los limites del mapa y con los vinilos, en los primeros dos casos la serpiente muere por lo que decimos que `self.alive = False`, pero si recolecta un vinilo, destruimos el vinilo, aumentamos en 1 la cantidad de vinilos que hemos consumido (`self.objective.rare += 1`) e invocamos el metodo `growth()`, pero, si el vinilo recolectado era un *vinilo dorado*, entonces aumentaremos el contador `self.weirdTimer` en 200; y por ultimo tenemos el metodo `move()` que mueve cada parte de la serpiente.

`camera.py`

- `Camera()`: Esta clase se inicializa con la serpiente a la que va a seguir nuestra camara (`self.snakeView`), y 3 bool (`self.topView`, `self.lolView` y `self.headView`), donde solo uno puede ser `True` a la vez, y el que sea `True` indicará cual sera la camara que tendremos, donde `self.topView = True` nos da una vista desde arriba, `self.lolView = True` nos dará una vista desde arriba en diagonal y por último `self.headView = True` nos dará una vista en 3ra persona desde atrás de la cabeza de la serpiente.

Esta clase cuenta con un solo metodo, `view()`, que se encarga de realizar las transformaciones para que la camara se ubique en el lugar correcto con la orientación correcta según indiquen los bool que mencionamos anteriormente.

`viny1.py`

- `Vinyl()`: Esta clase se inicializa con la posición del vinilo en el centro del mapa (`self.x = 0.0, self.y = 0.0, self.z = -4.5`), el estado del vinilo (si existe en el mapa: `self.exists = True`, y si no, `self.exists = False`), el ángulo de rotación de su animación (`self.theta`), un contador (`self.rare`) que al llegar a un multiplo de 5 reemplazará el vinilo con un *vinilo dorado* que al ser recolectado por SNOK provocará cambios de luces en la escena, una lista de 3-tuplas (`self.bans`) que contiene las posiciones que están ocupadas por la serpiente, el dibujo del vinilo (`self.GPU`), y la transformacion del dibujo del vinilo (`self.transform`).

Además esta clase cuenta con el metodo `spawn()` que hace que el vinilo se coloque en una posición aleatoria del mapa siempre que esta no este ocupada por la serpiente, y uno vez lo posiciona hace que “exista” (`self.exist = True`) el vinilo; además tenemos el metodo `posChecker()` que se encarga de revisar si la posición en la que se va a colocar el vinilo está ocupada o no; tenemos el metodo `draw()` que dibuja al vinilo o al *vinilo dorado* dependiendo del valor de `self.rare`; y por último tenemos el método `update()` que actualiza la animacion del vinilo.

floor.py

- `Floor()`: Esta clase se inicializa con los 5 dibujos que hacen la animacion del piso (`self.GPUfloor1-self.GPUfloor5`), la transformacion del piso (`self.transform`), el color (en RGB) con que se iluminara el piso cuando se recolecte el *vinilo dorado* (`self.r`, `self.g`, `self.b`), un contador (`self.weirdTimer`) que dirá cuanto tiempo queda de la iluminación especial (estado posterior a recoger el *vinilo dorado*), un contador (`self.timer`) para saber cuando cambiar la animacion del piso, y un contador (`self.floorPicker`) que nos dirá que dibujo de la animacion corresponde dibujar.

Esta clase además cuenta con 2 metodos, `draw()` que está encargado de dibujar el el piso segün dicten los parametros `self.weirdTimer` y `self.floorPicker`, y `update()` que se encarga de cambiar el color (en RGB) durante el estado posterior a recoger un *vinilo dorado*, mediante la selección de valores aleatorios (entre 0 y 1) para `self.r`, `self.g` y `self.b`

screens.py

- `Screens()`: Esta clase se inicializa con un contador (`self.c`) para la animacion del texto en la pantalla principal, un contador (`self.d`) para la animacion del texto en la pantalla de game over, un bool (`self.go`) que dicta la dirección de la animacion del texto en la pantalla de inicio, un bool (`self.do`) que dicta la dirección de la animacion del texto en la pantalla de game over, el dibujo del titulo del juego (`self.GPUMenu`), el dibujo del mensaje que aparece debajo del titulo (`self.GPUUntitled`), el dibujo del texto GAME OVER (`self.GPUover`) y las transformaciones de los dibujos recien mencionados (`self.menuTransform`, `self.untitledTransform`, `self.overTransform`).

Esta clase cuenta ademas con 2 métodos, `mainMenu()` que dibuja el titulo del juego (`self.GPUMenu`) (y le aplica la animacion correspondiente), además de dibujar el mensaje “presiona la barra espaciadora para iniciar” (`self.GPUUntitled`) en la pantalla de inicio; y `gameOver()` dibuja (y anima) el mensaje “GAME OVER” en la pantalla de game over.

1.1.2. Controlador

El controlador se encarga de recibir el input del usuario, luego el controlador actualiza el modelo acorde a este input. En este caso el controlador se encarga de iniciar el juego, cambiar la dirección de Snake y cambiar la vista de la cámara. El controlador se inicializa con el jugador (`self.snake`) y la cámara (`self.camera`) como `None`. Esta clase tiene solo el método `on_key()` que se ejecuta al presionar una tecla con un resultado dependiente de la tecla presionada, presionar “**BARRA ESPACIADORA**” en el menu de inicio dara comienzo al juego, presionar “**A**” o “**D**” cambia la dirección de giro del jugador, y al presionar la tecla “**E**”, “**R**” o “**T**” cambiara la camara a una vista desde arriba, desde atras de la cabeza o desde arriba en diagonal respectivamente.

1.1.3. Vista

Se muestra la escena

Se crea una ventana de 1000×800 de nombre *DISCO SNOK! 3D* creamos el objeto controlador, creamos 4 pipelines, una para objetos 3D sin textura, sin iluminacion; una para objetos 3D con textura, sin iluminacion; una para objetos 3D con textura e iluminacion; y una para objetos 3D sin textura y sin iluminación, dejamos el fondo en negro, establecemos la proyeccion a utilizar, creamos los objetos floor (`Floor()`), vinyl (`Vinyl()`), snake (`Snake()`), screens (`Screens()`), camera (`Camera()`) y ademas le asignamos al controlador el jugador y la camara a controlar, a la serpiente le asignamos su objetivo y el piso, al vinilo le entregamos las posiciones de la serpiente, y a la camara le asignamos la cabeza de la serpiente para que siga; luego creamos los bordes del mapa, el edificio sobre el que estaremos jugando, y el fondo oscuro para señalar que el jugador cae si sale de los limites; se crean variables para medir el tiempo desde el comienzo del juego, medir los frames y actualizaciones por segundo (las últimas se crean solo con propósito de debugging), luego se comienza a reproducir la música y se entra al bucle del juego.

En el bucle, si el juego acaba de iniciar, se muestra la pantalla inicial, si se presionó la “**BARRA ESPACIADORA**” se avanza al juego y comienza el cronometro, se hacen calculos para obtener el Δt , se llama el metodo `spawn()` del objeto `vinyl` si es que no existe un vinilo en el mapa, cuando Δt es mayor a 1 se actualiza la animación del vinilo, la posicion de la serpiente, la animacion del piso, (y `floor.weirdTimer > 0` se cambian las luces), luego se ven las colisiones, se dibujan los limites del mapa el fondo y el edificio, y luego se dibuja la pista de baile, la serpiente, y el vinilo y si la serpiente no está viva (se muere) se dibuja la pantalla de game over y se anima.

2. Instrucciones de Ejecución

Se debe ejecutar el código `snok3D.py` para iniciar el juego (esto puede tardar un poco). En el código se utilizan las librerías: numpy 1.19.1, PyOpenGL 3.1.5 y glfw 1.12.0

2.1. Argumentos que recibe

No se recibe ningún argumento, solo es necesario ejecutar el código `snok3D.py`.

2.2. Teclas de control

- **Barra Espaciadora:** Inicia el juego
- **A:** Cambio dirección (izquierda)
- **D:** Cambio dirección (derecha)
- **R:** Cambio cámara (3ra persona desde atrás)
- **E:** Cambio cámara (vista desde arriba)
- **T:** Cambio cámara (vista desde arriba en diagonal)

3. Resultados

A continuación se adjuntan imagenes para asistir en la descripción del juego.



Figura 1: Pantalla inicial.

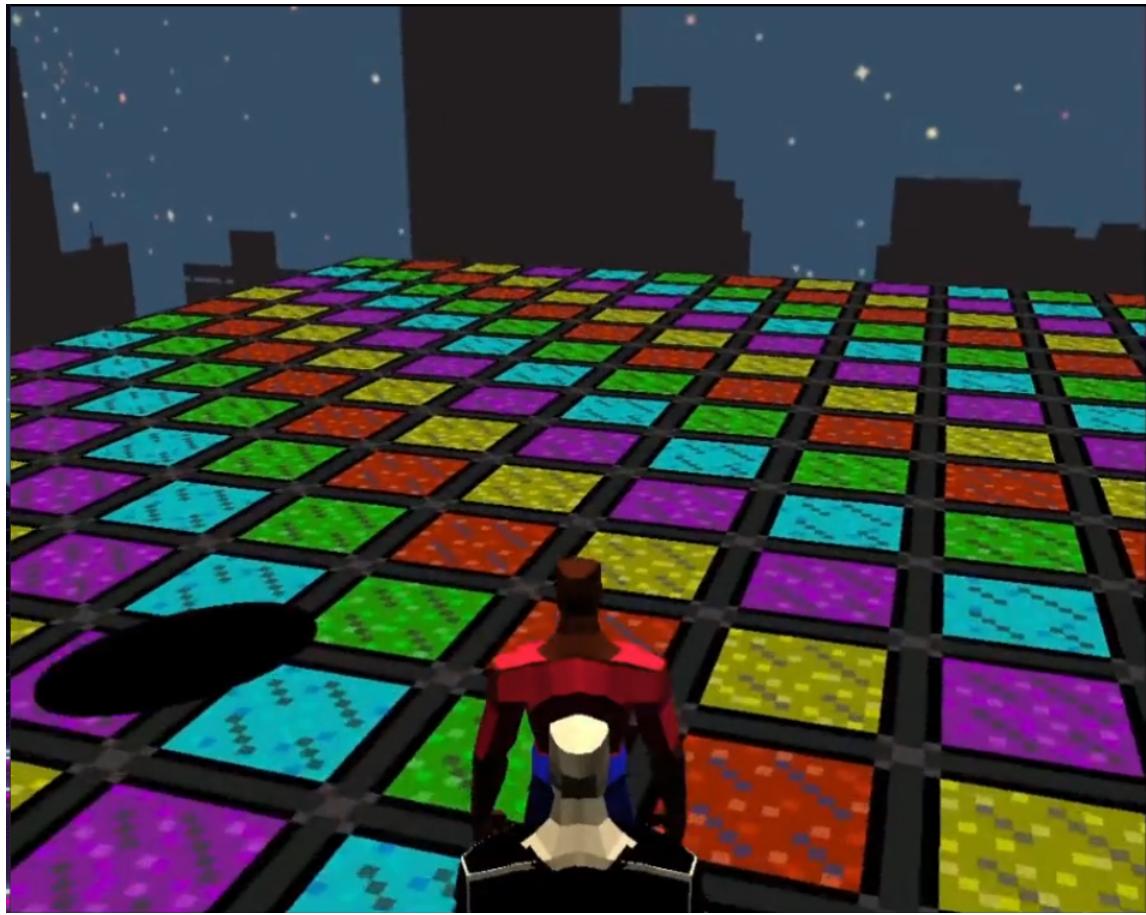


Figura 2: Pantalla de juego.(El jugador rojo (“Cabeza” de snok) a punto de recoger un vinilo)

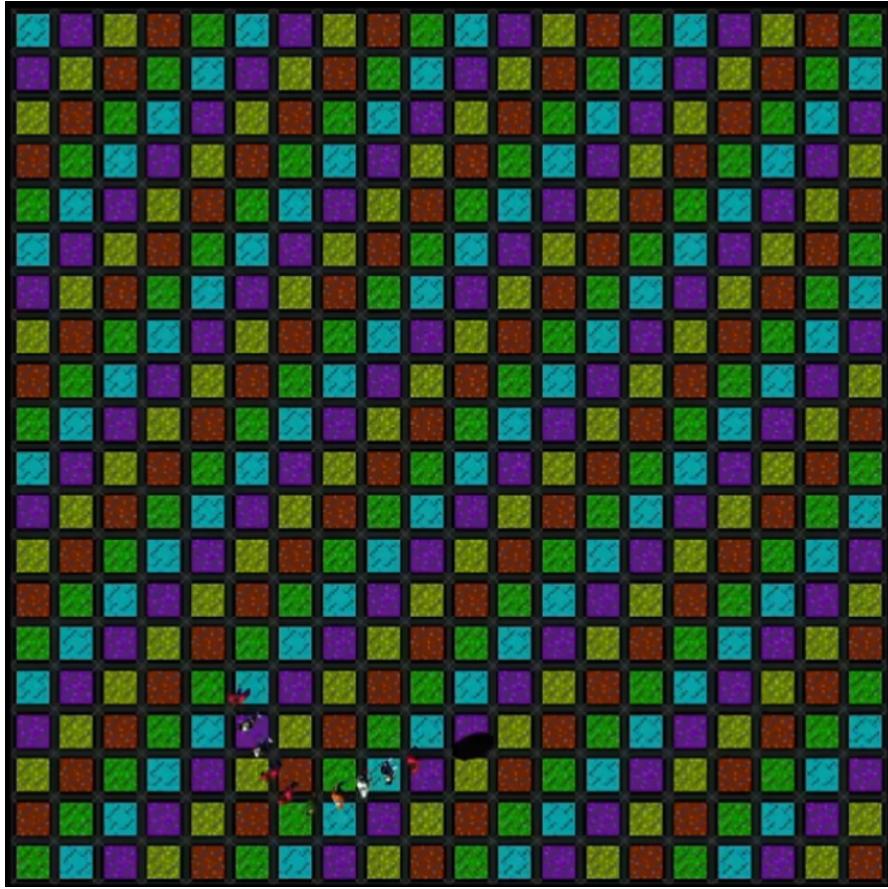


Figura 3: Pantalla de juego.(snok antes de recoger un vinilo)

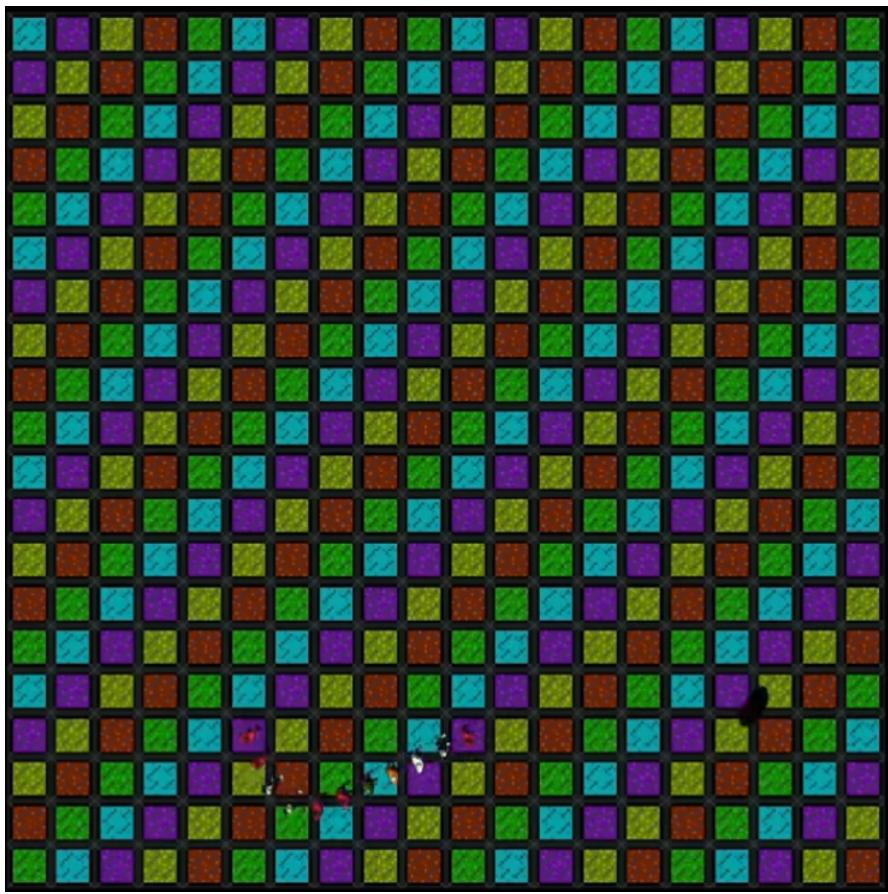


Figura 4: Pantalla de juego.(Crecimiento de snok luego de recoger un vinilo)



Figura 5: Pantalla de Game Over.(En medio de la animacion)