Información técnica

Animaciones

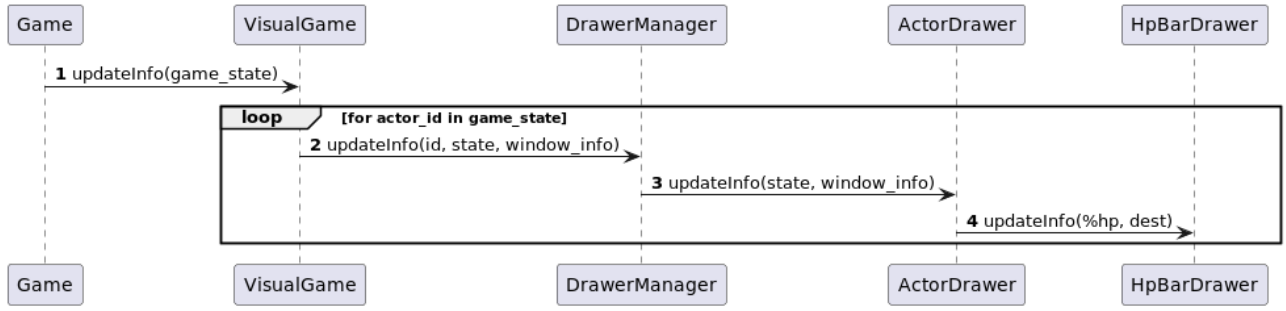
Las animaciones se cargan primero para luego dibujarse.

Las clases encargadas de cargar y preparar las texturas son las Animation.

Las clases encargadas de guardar un estado y luego dibujar son las Drawer.

Este estado se puede actualizar y tiene efectos al momento de dibujar.

Tener en cuenta que actualizar el estado y dibujar son metodos separados.

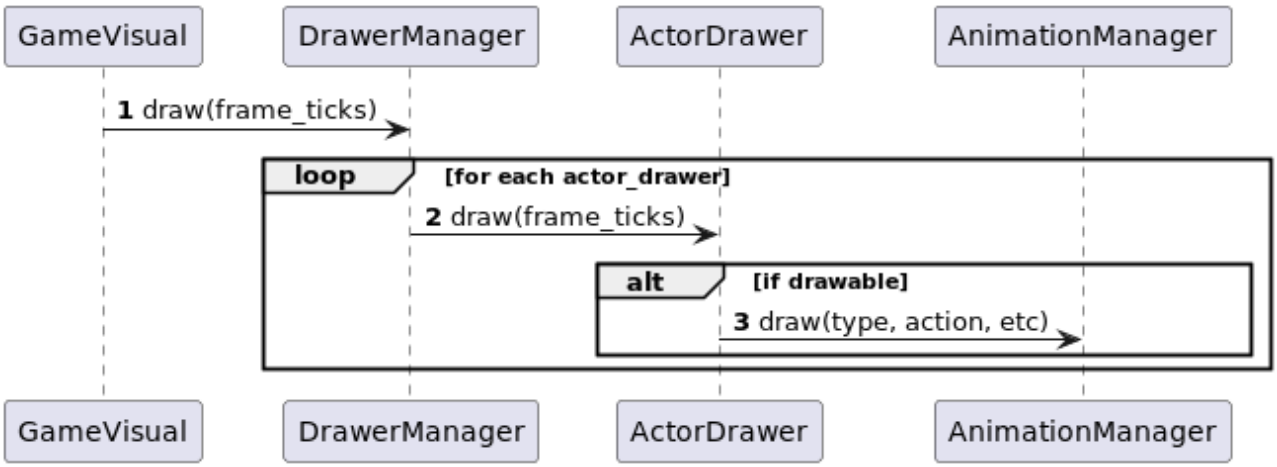


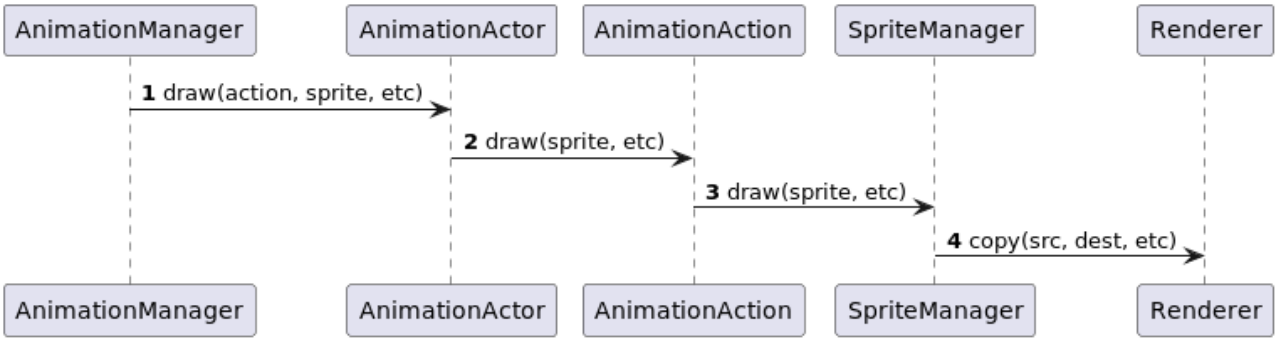
Actualizar la informacion determina la posicion del actor y de su barra de vida, el tipo de accion que debe dibujar, etc.

Al iniciar el programa primero se inicia el manejador de dibujadores quien instancia las animaciones.

Cuando se agregan nuevos actores se crean nuevos dibujadores que hacen uso de las animaciones ya creadas.

Las clases Drawer (dibujadores) delegan el dibujado a las Animation para respetar el encapsulamiento.





Un detalle es que el dibujador guarda el indice del sprite a dibujar pero el sprite manager es quien lo actualiza dependiendo de la velocidad de cada sprite, los milisegundos (frameticks) que pasaron y por cual frame va. Por ende se pasa la direccion de memoria de este indice y sprite manager recibe un puntero hacia ese indice.

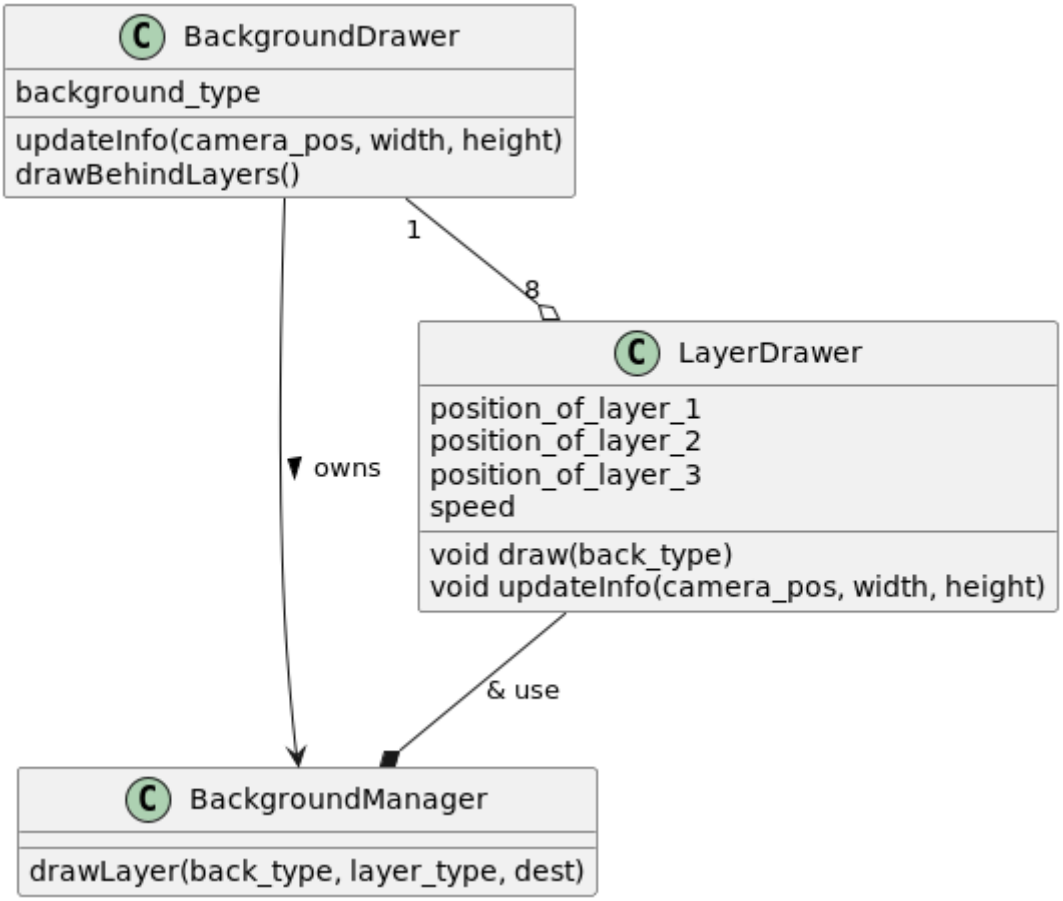
```
void SpriteManager::draw(SDL2pp::Texture &texture, std::uint8_t *sprite_index, std::uint8_t direction,
                         const SDL2pp::Point &sprite_destination, std::uint32_t frame_ticks) {
    if (frame_ticks > ms_to_change) {
        *sprite_index = loop_type.nextSprite(*sprite_index, max_index: sprites.size() - 1);
    }
```

```
void ActorDrawer::draw(std::uint32_t frame_ticks) {

    std::uint8_t last_sprite = sprite_index;

    if (!drawable)
        return;

    animation_manager.draw( actor_index: type, animation_index: animation, &sprite_index, direction,
                            sprite_destination, frame_ticks: frame_ticks - previous_frame_ticks);
```

El tipo de loop decide si la animacion se queda en el ultimo frame (por ejemplo, cuando un actor muere) o si vuelve al primero.

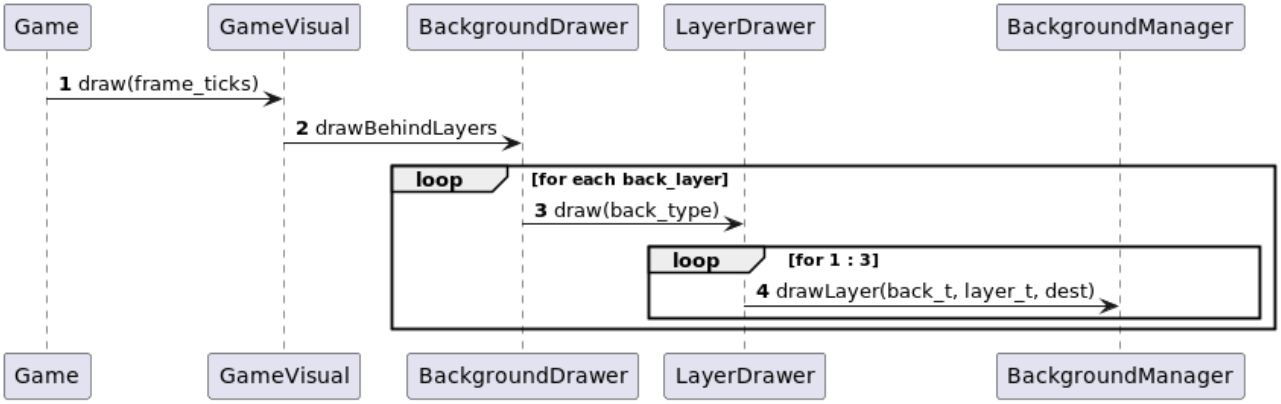
Background

Las texturas de los distintos layers del background son cargadas por el Background Manager el cual es inicializado por el Background Drawer. Este mismo tiene varios Layer Drawers que poseen la informacion necesaria para pedirle al Background Manager que los dibuje. Esto dado que Background Manager inicializa las texturas y es el propietario de las mismas.



Background Drawer actualiza su informacion de acuerdo al estado del juego (mas que nada la posicion del jugador) y cuando se llama al metodo draw() el mismo delega esta accion a todos los layers quienes luego delegan esta accion al Background Manager con la informacion del Layer a dibujar, la posicion, etc.

Se puede observar que este proceso es un poco similar a lo que ocurre con las animaciones de los actores.



Además, al dibujar un Layer se dibuja 2 copias más: una a la izquierda y otra a la derecha. Luego se mueven los 3 dibujos a la vez generando un background de mayor tamaño. Esto evita espacios vacios al moverse y brinda sensación de continuidad.

```
void LayerDrawer::draw(std::uint8_t background_type) {
    SDL2pp::Rect layer_1 = { x: x1,  y: 0,  w: width,  h: height};
    SDL2pp::Rect layer_2 = { x: x2,  y: 0,  w: width,  h: height};
    SDL2pp::Rect layer_3 = { x: x3,  y: 0,  w: width,  h: height};

    background_manager.drawLayer(background_type, layer_type,  destination: layer_1);
    background_manager.drawLayer(background_type, layer_type,  destination: layer_2);
    background_manager.drawLayer(background_type, layer_type,  destination: layer_3);
}
```