SDL 4 - Parallax + Animations

Ricard Pillosu - UPC

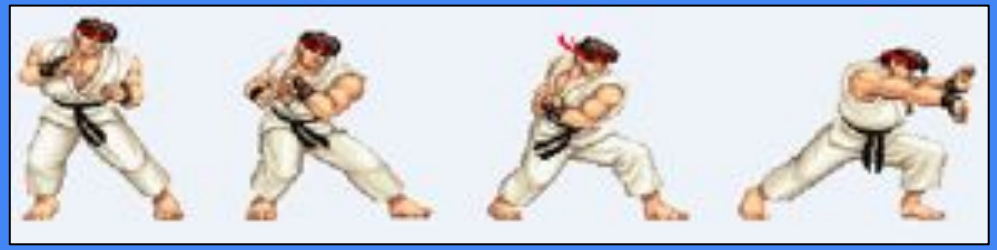
General improvements

- Now Module has a Start() method and virtual destructor:
  - `Module.cpp: bool Start () {return true;} + virtual ~Module() {}`
- New SCREEN_SIZE is a simple **scale** to resize small resolutions:
  - `Globals.h: #define SCREEN_SIZE 2`
- Now class p2Point included for player position:
  - `p2Point.h: iPoint position;`
- Get Familiar with SDL_Rect

Parallax!

- Trick to simulate depth in the map
- We now have a camera that will move around …
- But with a new parameter to Blit called “speed” (check it out)
- We simply adjust how camera movement affects each layer ...
- Resulting in a Parallax effect
- Most art was created just to fit the exact size of screen taking in account *“Parallax speed”*

Animation!



- Check Animation.h
- In the end, animation is a just a sequence of rectangles
- Implemented as a ring buffer with speed
- Very simplistic solution for now, it will get more complex later

```
Animation idle;
idle.PushBack({7, 14, 60, 90});
idle.PushBack({95, 15, 60, 89});
idle.PushBack({184, 14, 60, 90});
idle.PushBack({276, 11, 60, 93});
idle.PushBack({366, 12, 60, 92});
idle.speed = 0.2f;
```

Our first gameplay modules

- We introduce another round of execution: Start()
- ModulePlayer:
  - Responsible of managing the player and its actions.
  - Most of the code happens during Update()
- ModuleBackground:
  - Loads the assets and play background animations
  - Handles Parallax effect

TODO 1

“Make the camera move left and right”

- Follow the code around you
- This is our debug camera to check parallax planes
- While in game, we should never use it

TODO 2

“Draw the ship from the sprite sheet with some parallax effect”

- You will need a simple graphics App like Paint
- Find the coordinates for the ship in the background and draw it
- Take in account Parallax effect, give it some speed value (video)
- Make the ship move up and down a little to simulate buoyancy

TODO 3

“Animate the girl on the ship (see the sprite sheet)”

- Again you first need to gather all coordinates
- Follow the code that animates the flag in the background
- Mind that *Animation::GetCurrentFrame()* returns you the current frame **and** moves forward the frame count.
- If your ship is moving up & down you will need to take it in account.

TODO 4

“Make ryu walk backwards with the correct animations”

- Follow the code that animates him walking forward
- Find out the coordinates and play the animation while moving
- This time you need to update the position too
- The correct mix of movement speed and animation is very important!

Homework

Have Ryu:

- Throw a punch
- Kick
- Jump upwards (not diagonal): *you will need to update its position carefully!*