Tema 5: Arquitectura de un videojuego - Parte 1

Tecnología de la Programación de Videojuegos 1

Grado en Desarrollo de Videojuegos

Curso 2023-2024

Miguel Gómez-Zamalloa Gil con cambios de Rubén Rubio Cuéllar

Departamento de Sistemas Informáticos y Computación

Universidad Complutense de Madrid

Introducción

- En este tema proponemos una arquitectura básica para desarrollar videojuegos sencillos
  - Basada en programación orientada a objetos
  - Usando la biblioteca SDL
  - Proporciona un diseño composicional en el que resulta sencillo extender o actualizar el juego en distintas dimensiones

- La arquitectura se irá refinando según avance el curso:
  - Parte 1: Clases `Game`, `GameObject` (por ejemplo `Dog` para el ejerc. 3 del tema 3) y `Texture` (aún sin herencia)
  - Parte 2: Jerarquía de herencia de objetos del juego y aprovechando el polimorfismo
  - Parte 3: Estados del juego y callbacks
  - Parte 4: Patrones, componentes, etc. (en TPV2, 2º cuatrimestre)

La clase Texture

Texture.h

```
class Texture {
private:
   SDL_Texture* texture = nullptr;
   SDL_Renderer* renderer = nullptr;
   uint w = 0; uint h = 0; // Texture width and height
   uint fw = 0; uint fh = 0; // Frame width and height
   uint numCols = 1; uint numRows = 1;
public:
   Texture(SDL_Renderer* r) : renderer(r){};
   Texture(SDL_Renderer* r, string filename, uint numRows = 1, uint numCols = 1)
      : renderer(r) { load(filename, numRows, numCols); };
   ~Texture(){ free(); };
   void free();
   int getW() const { return w; };
   ...
   void load(string filename, uint numRows = 1, uint numCols = 1);
   void render(const SDL_Rect& rect, SDL_RendererFlip flip = SDL_FLIP_NONE) const;
   void renderFrame(const SDL_Rect& destRect, int row, int col, int angle = 0,
   SDL_RendererFlip flip = SDL_FLIP_NONE) const;
};
```

La clase Texture

Texture.cpp

```
#include "Texture.h"
void Texture::free(){
   SDL_DestroyTexture(texture);
   texture = nullptr;
   w = h = 0;
}
void Texture::load(string filename, uint nRows, uint nCols) {
   SDL_Surface* tempSurface = IMG_Load(filename.c_str());
   if (tempSurface == nullptr)
      throw "Error loading surface from " + filename;
   free();
   texture = SDL_CreateTextureFromSurface(renderer, tempSurface);
   if (texture == nullptr)
      throw "Error loading texture from " + filename;
   numRows = nRows;
   numCols = nCols;
   w = tempSurface->w; h = tempSurface->h;
   fw = w / numCols; fh = h / numRows;
   SDL_FreeSurface(tempSurface);
}
...
```

La clase Texture

Texture.cpp

```
...
void Texture::render(const SDL_Rect& destRect,
                     SDL_RendererFlip flip) const {
   SDL_Rect srcRect;
   srcRect.x = 0; srcRect.y = 0;
   srcRect.w = w; srcRect.h = h;
   SDL_RenderCopyEx(renderer, texture, &srcRect, &destRect, 0, 0, flip);
}
void Texture::renderFrame(const SDL_Rect& destRect, int row, int col, int angle,
                          SDL_RendererFlip flip) const {
   SDL_Rect srcRect;
   srcRect.x = fw * col;
   srcRect.y = fh * row;
   srcRect.w = fw;
   srcRect.h = fh;
   SDL_RenderCopyEx(renderer, texture, &srcRect, &destRect, angle, 0, flip);
}
```

La clase Game

Game.h

```
constexpr uint WIN_WIDTH = 800;
constexpr uint WIN_HEIGHT = 600;
constexpr uint NUM_TEXTURES = 3;
...
class Game {
private:
    SDL_Window* window = nullptr;
    SDL_Renderer* renderer = nullptr;
    // uint winWidth, winHeight; // También podrían estar aquí
    Dog* dog = nullptr;
    Helicopter* helicopter = nullptr;
    bool exit = false;
    Texture* textures[NUM_TEXTURES];
public:
    Game();
    ~Game();
    void run();
    void render() const;
    void handleEvents();
    void update();
};
```

La clase Game

Game.cpp

```
Game::Game() {
    // We first initialize SDL
    SDL_Init(SDL_INIT_EVERYTHING);
    window = SDL_CreateWindow("...", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,
                              WIN_WIDTH, WIN_HEIGHT, SDL_WINDOW_SHOWN);
    renderer = SDL_CreateRenderer(window, -1, SDL_RENDERER_ACCELERATED);
    if (window == nullptr || renderer == nullptr)
        throw "Error loading SDL window or renderer"s;
    // We now create the textures
    for (uint i = 0; i < NUM_TEXTURES; i++) {
        ...
    }
    // We finally create the game objects
    dog = new Dog(...);
    helicopter = new Helicopter(...);
}
Game::~Game() {
    for (uint i = 0; i < NUM_TEXTURES; i++) delete textures[i];
    SDL_DestroyRenderer(renderer);
    SDL_DestroyWindow(window);
    SDL_Quit();
}
```

La clase Game

Game.cpp

```
void Game::run() {
   while (!exit) { // Falta el control de tiempo
      handleEvents();
      update();
      render();
   }
}
void Game::update(){
   dog->update();
   ...
}
void Game::render() const {
   SDL_RenderClear(renderer);
   dog->render();

   SDL_RenderPresent(renderer);
}
void Game::handleEvents() {
   SDL_Event event;
   while (SDL_PollEvent(&event) && !exit) {
      if (event.type == SDL_QUIT) exit = true;
      dog->handleEvents(event);
      ...
   }
}
```

Ejemplo de clase GameObject (Dog)

Dog.h

```
class Dog {
private:
   uint w = 0; // width
   uint h = 0; // height
   uint x = 0; uint y = 0; // Posición de esquina superior izqda
   int dirX = 0; int dirY = 0; // Dirección de movimiento
   Texture* texture = nullptr;
public:
   Dog(){}
   Dog(uint w, uint h, uint x, uint y, Texture* t) :
      w(w), h(h), x(x), y(y), texture(t) {}
   ~Dog() {}
   void render() const;
   void update();
   void handleEvents(SDL_Event& event);
};
```