Programiranje 2

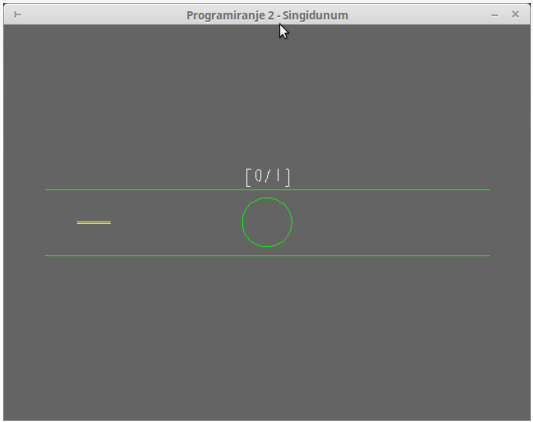
Klase primer kroz igru

dr Đorđe Obradović

Singidunum - Centar Novi Sad

Igra 01

Jednostavna igra implementirane u SDL



Opis i cilj igre

Dve žute linije se kreću horizontalno konstantnom brzinom.

Pritiskom na [SPACE] žute linije se zaustavljaju.

Ako se zaustave unutar zelenog kruga score se uvećava za 1.

Cilj je imati što veći odnos 'ubačenih' i pokušanih.

Implementacija - SDL

Uvod

Simple DirectMedia Layer

Platforma koja nezavisno od operativnog sistema omogućuje pristup:

- audio
- keyboard
- mouse
- joystick
- grafika preko OpenGL ili Direct3D

Podrška

- Windows
- Mac OS X
- Linux
- iOS
- Android

SDL - osnova

Biblioteke

```
0  #include <SDL2/SDL.h>
1  #include <SDL2/SDL_ttf.h>
2  #include <vector>
3  #include <math.h>
4  #include <iostream>
5  #include <sstream>
6  using namespace std;
```

Main funkcija

```
0   int main(int argc, char ** argv) {
1       bool quit = false;
2       SDL_Event event;
3
4       // init SDL ================================================
5       SDL_Init(SDL_INIT_VIDEO);
6       SDL_Window * window = SDL_CreateWindow("Programiranje 2 - Singidunum",
7           SDL_WINDOWPOS_UNDEFINED, SDL_WINDOWPOS_UNDEFINED, 640, 480, 0);
8       SDL_Renderer * renderer = SDL_CreateRenderer(window, -1, 0);
9
10      int ret = TTF_Init();
11      TTF_Font* font = TTF_OpenFont("lazy.ttf", 24);
12
13      // main loop ===============================================
14
15      // cleanup SDL =============================================
16      SDL_DestroyRenderer(renderer);
17      SDL_DestroyWindow(window);
18      TTF_Quit();
19      SDL_Quit();
20      return 0;
21  }
```

Main loop

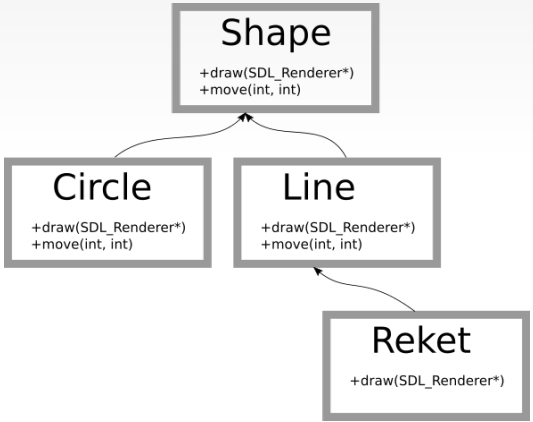
```
0  while (!quit){
1      SDL_Delay(2);
2      SDL_PollEvent(&event);
3      // event handling ---------------------------------
4
5      // clear window-----------------------------------
6      SDL_SetRenderDrawColor(renderer, 100, 100, 100, 255);
7      SDL_RenderClear(renderer);
8      drawScore(score, total, font, renderer);
9
10     // render objects---------------------------------
11     SDL_SetRenderDrawColor(renderer, 0, 255, 0, 255);
12
13     // render window----------------------------------
14     SDL_RenderPresent(renderer);
15 }
```

Event handling

```
0   switch (event.type){
1     case SDL_QUIT:
2         quit = true;
3         break;
4     case SDL_KEYDOWN: //SDL_KEYUP:
5         switch( event.key.keysym.sym ){
6             case SDLK_LEFT:
7                 break;
8             case SDLK_RIGHT:
9                 break;
10            case SDLK_UP:
11                break;
12            case SDLK_DOWN:
13                break;
14            case SDLK_r:
15                break;
16            case SDLK_ESCAPE:
17                quit = true;
18                break;
19            case SDLK_SPACE:
20                break;
21            default:
22                break;
23        }
24        break;
25  }
```

Igra

Dijagram klasa



Shape

```
0  class Shape{
1    public:
2      virtual void draw(SDL_Renderer* renderer){
3        SDL_RenderDrawLine(renderer, 20, 40, 60, 40);
4        SDL_RenderDrawLine(renderer, 40, 20, 40, 60);
5      };
6
7      virtual void move(int, int){};
8  };
```

Line - deklaracija

```
0  class Line:public Shape{
1    public:
2      Line(int, int, int, int);
3      int x1, y1, x2, y2;
4      void draw(SDL_Renderer*);
5      void move(int, int);
6  };
```

Line - konstruktor

```
0  Line::Line(int aX1, int aY1, int aX2, int aY2){
1    x1 = aX1;
2    y1 = aY1;
3    x2 = aX2;
4    y2 = aY2;
5  }
```

Line - draw funkcija

```
0  void Line::draw(SDL_Renderer* renderer){
1    SDL_RenderDrawLine(renderer, x1, y1, x2, y2);
2  }
```

Line - move funkcija

```
0  void Line::move(int dX, int dY){
1    x1 += dX;
2    x2 += dX;
3    y1 += dY;
4    y2 += dY;
5  }
```

Circle - deklaracija

```
0  class Circle:public Shape{
1    public:
2      Circle(int, int, int);
3      int xc;
4      int yc;
5      int r;
6      void draw(SDL_Renderer*);
7      void move(int, int);
8  };
```

Circle - konstruktor

```
0  Circle::Circle(int aXc, int aYc, int aR){
1      xc = aXc;
2      yc = aYc;
3      r = aR;
4  }
```

Circle - draw

```
0   void Circle::draw(SDL_Renderer* renderer){
1     int s = 30;
2     float d_a = 2*M_PI/s;
3     float angle = d_a;
4
5     float x0, y0;
6     float x1, y1;
7     x1 = xc+r;
8     y1 = yc;
9     for (int i=0; i<s; i++){
10      x0 = x1;
11      y0 = y1;
12      x1 = xc + cos(angle) * r;
13      y1 = yc + sin(angle) * r;
14      angle += d_a;
15      SDL_RenderDrawLine(renderer, x0, y0, x1, y1);
16    }
17  }
```

Circle - move

```
0  void Circle::move(int dX, int dY){
1    xc += dX;
2    yc += dY;
3  }
```

Reket - deklaracija

```
0  class Reket:public Line{
1    public:
2      Reket(int x1, int y1, int x2, int y2):Line(x1, y1, x2, y2){};
3      void draw(SDL_Renderer*);
4  };
```

Reket - draw

```
0  void Reket::draw(SDL_Renderer* renderer){
1    SDL_RenderDrawLine(renderer, x1, y1-1, x2, y2-1);
2    SDL_RenderDrawLine(renderer, x1, y1+1, x2, y2+1);
3  }
```

Ispis rezultata

```
0   void drawScore(int score, int total, TTF_Font* font, SDL_Renderer* renderer){
1     SDL_Color white = {255, 255, 255};
2     stringstream ss;
3     ss << "[" << score<<"/"<<total << "]";
4     SDL_Surface* sm = TTF_RenderText_Solid(font, ss.str().c_str(), white);
5     SDL_Texture* poruka = SDL_CreateTextureFromSurface(renderer, sm);
6     SDL_Rect poruka_box; //create a rect
7     poruka_box.x = 320-sm->w/2;
8     poruka_box.y = 200-2-sm->h;
9     poruka_box.w = sm->w;
10    poruka_box.h = sm->h;
11    SDL_RenderCopy(renderer, poruka, NULL, &poruka_box);
12  }
```

Inicijalizacija objekata

```
0   Reket* reket = new Reket(320-20, 240, 320+20, 240);
1
2   vector<Shape*> lista;
3   lista.push_back(new Line(50, 200, 640-50, 200));
4   lista.push_back(new Circle(320, 240, 30));
5   lista.push_back(new Line(50, 280, 640-50, 280));
```

Crtanje objekata

```
0   SDL_SetRenderDrawColor(renderer, 0, 255, 0, 255);
1   for(int i=0; i<n; i++){
2     lista[i]->draw(renderer);
3   }
4   SDL_SetRenderDrawColor(renderer, 255, 255, 0, 100);
5   reket->draw(renderer);
```

Literatura

Literatura

- https://www.libsdl.org/
- https://en.wikipedia.org/wiki/Simple_DirectMedia_Layer
- http://lazyfoo.net/tutorials/SDL/

Korisno Ubuntu

```
sudo apt-get install libsdl2-dev
sudo apt-get install libsdl2-image-2.0-0
sudo apt-get install libsdl2-image-dev
sudo apt-get install libsdl2-image-dbg
sudo apt-get install libsdl2-ttf-dev
```

Domaći

Zadaci za vežbu

1. Nacrtati reket u obliku jednakostraničnog trougla čija je osnova linija širine trenutnog reketa, visina je 5 pixela.
2. Ubaciti da se meta (zeleni krug) prikazuje kao kvadrat čije stranice su jednake prečniku kruga iz primera i plave je boje.
3. Dodati pomeranje mete (zelenog kruga ili plavog kvadrata)
4. Kretanje centra reketa po dijagonali unesto horizontalno
5. Kretanje centra reketa po kvadratnoj funkciji umesto horizontalno.
6. Implementirati postepeno ubrzavanje kretanja žutih linija.