Информатика. Семинар 4

15 марта 2016 года

https://dl.dropboxusercontent.com/u/96739039/sem4/infa_s04.pdf

Мы уже умеем…

- Перемещать персонажа по экрану, не давая ему выйти за пределы

Как задать фон?

```
sf::RenderWindow window(sf::VideoMode(800, 600), "My window");
window.setFramerateLimit(60);

sf::Texture mapTexture;
if (!mapTexture.loadFromFile("political-map.png"))
{
  std::cout << "Can`t load texture\n";
}
mapTexture.setSmooth(true);
sf::Sprite background;
background.setTexture(mapTexture);
```

```
window.draw(background);
```

Как быть, если «мир» очень большой…

```
sf::View gameView(sf::FloatRect(0.0f, 0.0f,
  mapTexture.getSize().x, mapTexture.getSize().y));
window.setView(gameView);
sf::View initialView = gameView;

float scale = 1.0f;
```

Вне основного цикла

```
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Add) && scale < 1.0f)
{
  scale *= 1.03f;
}
if (sf::Keyboard::isKeyPressed(sf::Keyboard::Subtract))
{
  scale /= 1.03f;
}
            gameView = initialView;
            gameView.zoom(scale);
            window.setView(gameView);
```

Камера должна двигаться за героем…

```
Vector2 FindViewCenter(const sf::Sprite& hero,
  const sf::Sprite& background, sf::View& gameView)
{
  Vector2 center = Vector2(hero.getPosition()) +
    Vector2(hero.getLocalBounds().width / 2, hero.getLocalBounds().height / 2);
  center.x = std::max(center.x, gameView.getSize().x / 2);
  center.y = std::max(center.y, gameView.getSize().y / 2);

  center.x = std::min(center.x,
    background.getTexture()->getSize().x - gameView.getSize().x / 2);
  center.y = std::min(center.y,
    background.getTexture()->getSize().y - gameView.getSize().y / 2);
  return center;
}
```

```
Vector2 center = FindViewCenter(hero, background, gameView);
gameView.setCenter(sf::Vector2f(center.x, center.y));

window.setView(gameView);
```

setView вызываем один раз за основной цикл

Эффект плавного движения камеры

```
float scale = 1.0f;
float currScale = scale;
```

```
currScale = scale + (currScale - scale) * exp(-3e+0f * dt.asSeconds());
gameView = initialView;
gameView.zoom(currScale);
```

Упражнение 1: по аналогии реализовать плавное изменение положения камеры.

Как ещё можно задавать массивы…

```
#include <vector>

std::vector<Particle> particles;

Particle p;
p.pos = Vector2(0, 0);
p.velocity = Vector2(1, 1);
p.m = 1.0f;

particles.push_back(p);

for (size_t i = 0; i < particles.size(); ++i)
{
  particles[i].Update(...);
}
```

Удаление элемента их vector’a

```
particles.erase(particles.begin() + index);
```

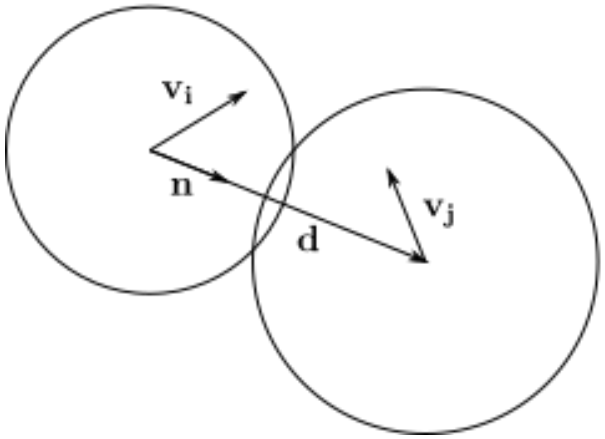
Перевод локальных координат экрана в глобальные

```
// get the current mouse position in the window
sf::Vector2i pixelPos = sf::Mouse::getPosition(window);
// convert it to world coordinates
sf::Vector2f worldPos = window.mapPixelToCoords(pixelPos);
```

Упражнение 2

- В месте нажатия кнопки мыши, создаём новую particle, и помещаем в вектор particles.
- Новой частице задаём некоторую скорость в произвольном направлении.
- Реализовать упругое отражение от границ «мира»
- Реализовать упругое столкновение между частицами

Алгоритм столкновения частиц



1. $|\mathbf{d}| \leq r_i + r_j$
2. $(\mathbf{v_j} - \mathbf{v_i}) \cdot \mathbf{d} \leq 0$
3. $\mathbf{\Delta p} = \left(2\left(\frac{\mathbf{v_j}-\mathbf{v_i}}{\frac{1}{m_i}+\frac{1}{m_j}}\right)\cdot\mathbf{n}\right)\mathbf{n}$
4. $\mathbf{v_j'} = \mathbf{v_j} + \frac{\mathbf{\Delta p}}{m_j} \quad \mathbf{v_i'} = \mathbf{v_i} - \frac{\mathbf{\Delta p}}{m_i}$

Неупругое столкновение: 2 в формуле для $\Delta\vec{p}$ заменить на $(1 + bounceFactor),\ bounceFactor \in [0 \ldots 1]$

Необходимые классы

```
class MaterialPoint
{
public:
  MaterialPoint();

  virtual ~MaterialPoint();
  virtual void UpdatePosition(float dt);

  Vector2 GetImpulse() const;

  Vector2 position;
  Vector2 velocity;
  Vector2 acceleration;

  float mass;
};
```

```
#include "MaterialPoint.h"

class Ball: public MaterialPoint
{
public:
  float radius;
  int type;
};
```