



# ДИПЛОМНА РОБОТА

на тему: "Генерація та відображення 3D-моделей об'єктів у комп'ютерних іграх з використанням сплайнів"

Виконав:

студент групи ПА-17-2

Панасенко Єгор Сергійович

Керівник:

Степанова Наталія Іванівна



# Постановка задачі


- дослідити математичні моделі просторових об'єктів;
- обрати оптимальний спосіб подання інформації про об'єкти, які потрібно відобразити;
- сформулювати вимоги до програми відображення 3D-об'єктів і виконати програмну реалізацію;
- розробити інтерфейс для створення та редагування 3D-об'єктів;
- зробити програмне забезпечення придатним для компіляції та роботи у різних операційних системах;
- розробити шейдер для генерації 3D-моделі за допомогою відеокарти;
- надати опис розробленого програмного забезпечення та створити схеми взаємодії його компонентів.



## Актуальність теми

У сучасному світі спостерігається неймовірний приріст потужності обчислювальної техніки і розробники ігор намагаються використати цю потужність найбільш ефективно з метою отримання графіки, найбільш схожої на реальний світ.

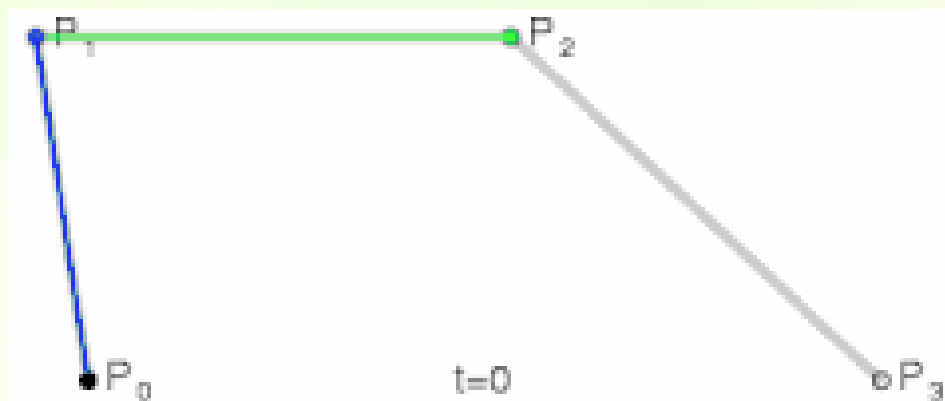
Для досягнення найбільшого задоволення розробники також намагаються створити якомога більше ігрових об'єктів та приголомшливих ефектів, що супроводжується значним споживання дискового простору й оперативної пам'яті. Тому завжди є актуальним питання розробки більш ефективних методів моделювання ігрових об'єктів, які б використовували менше обчислювальних ресурсів.





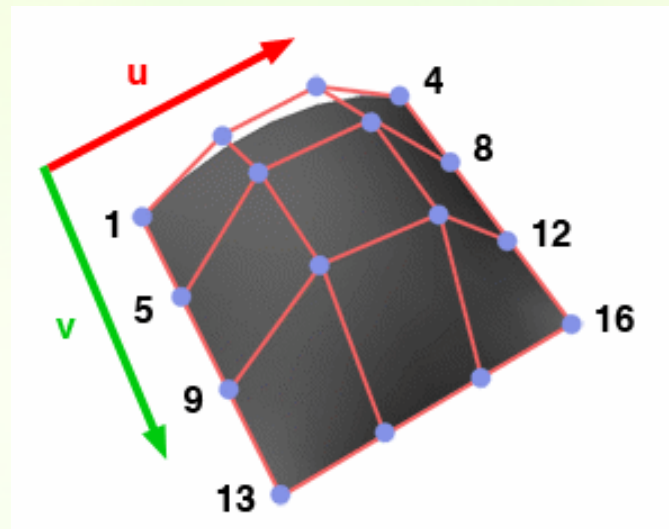
# Крива Безьє

$$\mathbf{B}(t) = (1-t)^3\mathbf{P}_0 + 3(1-t)^2t\mathbf{P}_1 + 3(1-t)t^2\mathbf{P}_2 + t^3\mathbf{P}_3, 0 \leq t \leq 1.$$



# Поверхня Безьє

$$\mathbf{p}(u, v) = \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) \mathbf{k}_{i,j} \quad B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i} \quad \binom{n}{i} = \frac{n!}{i!(n-i)!}$$





Menu

Scenes

Scene 0

Textures

Open texture

None

Texture: "/home/gau

Models

Next Add cube

None

Model: "../share

Subwindow params 1

Wireframe

Light

Culling

Transparency

Camera mode

Textured

Pre generated

8

8

Level

Camera

0.000

0.000

-17.000

Position

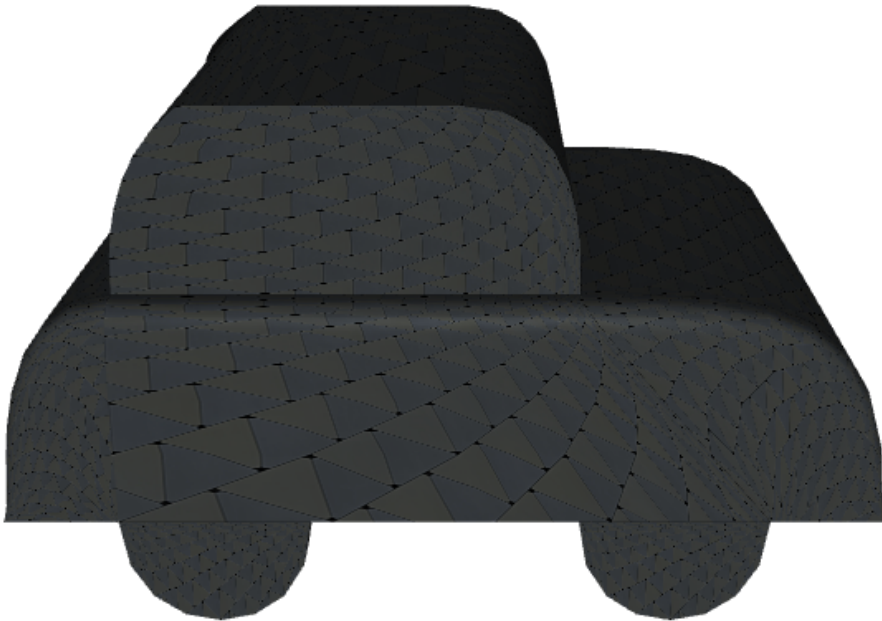
-0.000

0.000

-0.000

Rotation

Subwindow 1









Menu

Scenes

Scene 0

Textures

Open texture

None

Texture: "/home/gau

Models

Next Add cube

None

Model: "../share

Subwindow params 1

☒ Wireframe

☒ Light

☐ Culling

☐ Transparency

☐ Camera mode

☒ Textured

☐ Pre generated

64

64

Level

Camera

0.000

0.000

-17.000

Position

-0.000

0.000

-0.000

Rotation

Subwindow 1





Menu

Scenes

Scene 0

Textures

Open texture

None

Texture: "/home/gau

Models

Next

Add cube

None

Model: "../share

Subwindow params 1 Bezier mesh

Regenerate

Save

0.300	0.000	-0.700	0
0.300	0.400	-0.700	1
0.800	0.400	-0.700	2
0.800	0.000	-0.700	3
0.000	0.400	-0.700	4
0.300	0.400	-0.700	5
1.500	0.400	-0.700	6
2.000	0.400	-0.700	7
2.300	0.400	-0.700	8
2.000	0.400	-0.700	9
1.500	0.400	-0.700	10

0

Invert light normals

1	1	2	2
1	1	2	2
1	1	2	2
1	0	3	2

1

Invert light normals

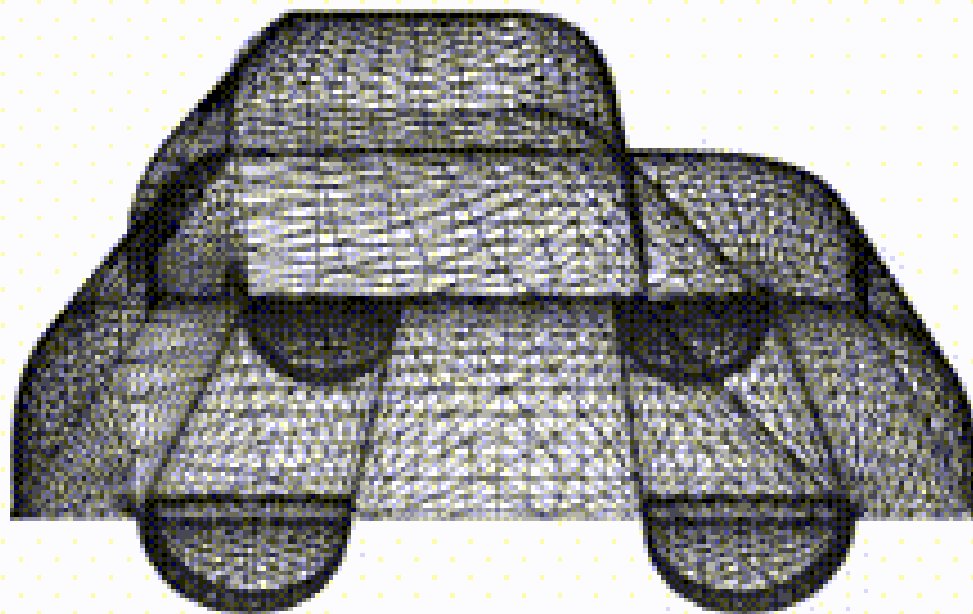
54	54	54	55
54	54	54	55
54	54	54	55
27	27	27	28

Subwindow 1





# Динамічна генерація мешу



# Документація коду

## 4 Алфавітний покажчик класів

### 4.1 Класи

#### Image

Зберігає масив пікселів, ширину та висоту

57

#### Mesh

Меш, який збе

7.3 Клас Buffer

60

7

#### Model

Інтерфейс до

Буфер, у якому відбувається рендеринг у текстурі.

#include <buffer.h>

Діаграма зв'язків класу Buffer:

65

70pi

14

#### Object

об'єкт моделі

67

18

#### Renderbuffer

Буфер рендер

78

22

#### Scene

Сцена із об'єк

82

32

#### Shader

Клас взаємодії

84

37

#### ShaderSource

95

46

#### Texture

Клас текстури

97

55

#### OSDO::vector< T

Вектор що не

Загальнодоступні елементи

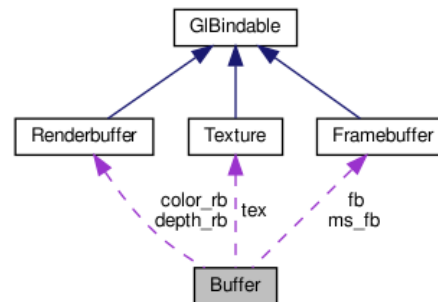
- bool [pre\\_render](#) (GLsizei size[2])  
Підготовка до рендерингу.
- void [post\\_render](#) (GLsizei size[2])  
Генерація текстури з утвореного кадру.
- const [Texture](#) & [get\\_tex](#) ()  
Забирає текстуру у яку проводиться рендеринг.

101

#### Vertex

Структура вер

109



# Висновки

- досліджено, які математичні моделі просторових об'єктів можуть використовуватися для нашої задачі;
- обрано поверхню Без'є, як оптимальний спосіб подання інформації про об'єкти, які потрібно відобразити;
- сформульовано вимоги до програми відображення 3D-об'єктів і виконано програмну реалізацію згідно цих вимог;
- розроблено інтерфейс для створення та редагування 3D-об'єктів;
- програмне забезпечення зроблене придатним для компіляції та роботи у операційних системах Linux та Windows;
- розроблено теселяційний шейдер для генерації 3D-моделі за допомогою відеокарти;
- за допомогою інструменту Doxygen на основі коду програмного забезпечення надано його опис та створено схеми взаємодії його компонентів.



**Дякую за увагу!**

