ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

ФАКУЛТЕТ КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ

СПЕЦИАЛНОСТ КОМПЮТЪРНО И СОФТУЕРНО ИНЖЕНЕРСТВО

Студент: Валери Ивайлов Райков

Фак. №: 121222139

Група: 42Б

Компютърна графика

Курсов проект на тема 1:

Animated, textured robot, able to move using mouse and keys

Дата на предаване: 16/11/2025г.

Съдържание

Резюме

Обхват и цел на проекта:

Настоящият проект има за цел разработка на интерактивна 3D графика -приложение, реализиращо анимиран робот с пълно управление в триизмерно пространство. Проектът демонстрира практическо приложение на съвременни компютърно-графични техники, включвайки рендиране в реално време, различни по вид транслации, осветление, анимации и потребителски интерфейс [1]. Потребителят получава пълен контрол върху движението, анимациите и средата на робота, като по този начин се демонстрират ключови принципи на компютърната графика.

Изследователски проблем:

Проектът адресира проблема за създаване на интуитивен интерфейс за управление на сложна 3D анимация, комбинирайки математически трансформации, графично програмиране и потребителски взаимодействия [2].

1. **Математическо моделиране на 3D пространството:**

   - **Координатни системи:** Управление на световни, локални и координатни системи на камерата [3].
   - **Трансформации:** Прилагане на матрици за транслация, ротация и скалиране за позициониране на робота и неговите части (крайници) [4].
   - **Йерархични модели:** Роботът е йерархична структура (напр. "родител-дете"). Ротацията на рамото трябва да повлияе и на цялата ръка, която държи. Това изисква ефективно изчисляване на матриците на трансформация за всяка става в йерархията [5].

2. **Графично програмиране и рендиране:**

   - **OpenGL:** Избор и използване на модерна графична API за комуникация с GPU [6].
   - **Graphics Pipeline:** Програмиране на шейдъри – малки програми, работещи на видеокартата [7].
   - **Vertex Shader:** Отговаря за позициите на върховете и трансформациите.
   - **Fragment Shader:** Отговаря за оцветяването на всеки пиксел, включително ефекти от осветление, текстури и рефлективност.
   - **Буфери:** Управление на Vertex Buffer Objects (VBOs) и Vertex Array Objects (VAOs) за ефективно предаване на геометрични данни към GPU.

3. **Реализиране на интуитивен потребителски интерфейс (UI):**

Проблемът е да се "преведе" сложната 3D математика в прост и лесен за използване интерфейс [9]. Това може да включва:

- **Widget-и за трансформация:** Виртуални "контроли" (стрелки, кръгове) в 3D пространството, които позволяват плъзгане за преместване и ротация.
- **Панел с плъзгачи (Sliders):** За прецизен контрол на ъгъла на всяка става.
- **Дървовидна структура (Tree View):** За визуализация и селекция на отделните части на робота.
- **Контекстно меню:** За бърз достъп до често използвани действия (анимиране, промяна на атрибути, нулиране на поза и т.н.).

Въведение

Контекст:

Развитието на 3D графиката и интерактивните приложения представлява фундаментална област в съвременното компютърно програмиране [1]. Този проект се вписва в контекста на образователни и изследователски инициативи в областта на компютърната графика.

Обхват:

Проектът обхваща проектирането и реализацията на пълнофункционално приложение за 3D визуализация [10], включващо:

- Създаване на 3D модел на робот с множество части
- Имплементация на анимационна система
- Разработване на интерактивен потребителски интерфейс
- Реализиране на система за управление на камера и осветление

Цел:

Основната цел е създаването на образователна платформа, демонстрираща принципите на 3D графиката, трансформациите в пространството и интерактивното управление на сложни обекти [2].

Основна част

Методология

Проектът следва методологията на инкрементално разработване, като се започва от проста основа и постепенно се добавят функционалности:

Фаза 1: Настройка на графичния контекст и базово рендиране

Фаза 2: Създаване на основна геометрия (куб)

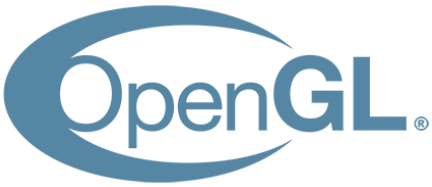
Фаза 3: Имплементация на трансформации и йерархия

Фаза 4: Добавяне на анимации и управление

Фаза 5: Интегриране на потребителски интерфейс

Използвани технологии

- **OpenGL и GLFW -** OpenGL предоставя API за графично рендиране, докато GLFW управлява прозорци и входни събития




```
// Инициализация на графичния контекст
GLFWwindow* init_window(void)
{
    glfwSetErrorCallback(glfw_error_callback);
```

```
    if (glfwInit() == GLFW_FALSE)
        return nullptr;

    glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR,cg::version.gl_major);
    glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR,cg::version.gl_minor);
    glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_CORE_PROFILE);
    glfwWindowHint(GLFW_OPENGL_FORWARD_COMPAT, GLFW_TRUE);
    glfwWindowHint(GLFW_SAMPLES, 4);
}
```

- **GLM (OpenGL Mathematics) -** предоставя математически функции специално разработени за компютърна графика.

// Матрични трансформации за робота

```
g_model = glm::translate(g_model, cg::robot.position);
g_model = glm::rotate(g_model, glm::radians(cg::robot.rotation.y),
glm::vec3(0.0f, 1.0f, 0.0f));
```

- **ImGui (Dear ImGui)**

// Създаване на контролен панел

```
ImGui::Begin("Robot Controls");
ImGui::Text("Rotation: (%.2f, %.2f, %.2f)",
ImGui::Separator();
ImGui::SliderFloat("Move Speed", &g_move_speed, 0.01f, 1.0f);
ImGui::Button("Reset Robot");
```

Архитектура на приложението

- **Модел на данните**

Проектът използва структурирани данни за управление на състоянието:

```
struct Robot {
      glm::vec3 position = glm::vec3(0.0f);
      glm::vec3 rotation = glm::vec3(0.0f);
      glm::vec3 scale = glm::vec3(1.0f);
}
```

- **Графичен пайплайн**

Vertex Shader: работи във веригата за рендиране (graphics pipeline). Неговата основна задача е да обработва отделните върхове (vertices) на 3D обектите.

```
glsl
#version 460 core
layout(location = 0) in vec3 a_pos;
uniform mat4 u_model;
uniform mat4 u_view;
uniform mat4 u_projection;

void main() {
    gl_Position = u_projection * u_view * u_model * vec4(a_pos, 1.0);
}
```

Fragment Shader: обработва отделните фрагменти (пиксели) в рамките на рендирания примитив.

```
glsl
#version 460 core
out vec4 frag_color;
uniform vec3 u_light_pos;
uniform vec3 u_light_color;

void main() {
    frag_color = vec4(final_color, 1.0);
}
```

Имплементация на анимациите

- Система за йерархични трансформации

```
// Функция за изчертаване на робота (неговите съставни части)
void draw_robot() {
    draw_cuboid(cg::robot.body_size);

    glm::mat4 body_model = g_model;
    g_model = glm::translate(g_model, glm::vec3(0.0f, 
cg::robot.body_size.y/2, 0.0f));
    draw_cuboid(cg::robot.shoulder_size);

    glm::mat4 shoulders_model = g_model;
    g_model = glm::translate(g_model, glm::vec3(-shoulder_width, 0.0f, 
0.0f));
    g_model = glm::rotate(g_model, glm::radians(cg::robot.arm_swing), 
glm::vec3(1.0f, 0.0f, 0.0f));
    draw_cuboid(cg::robot.arm_size);
}
```

- **Анимационни цикли**

```
static void render(void)
{
     static float time = 0.0f;
     time += 0.05f;

     cg::robot.arm_swing = sin(time * cg::robot.walk_speed) * 30.0f;
     cg::robot.leg_swing = sin(time * cg::robot.walk_speed) * 25.0f;
}
```

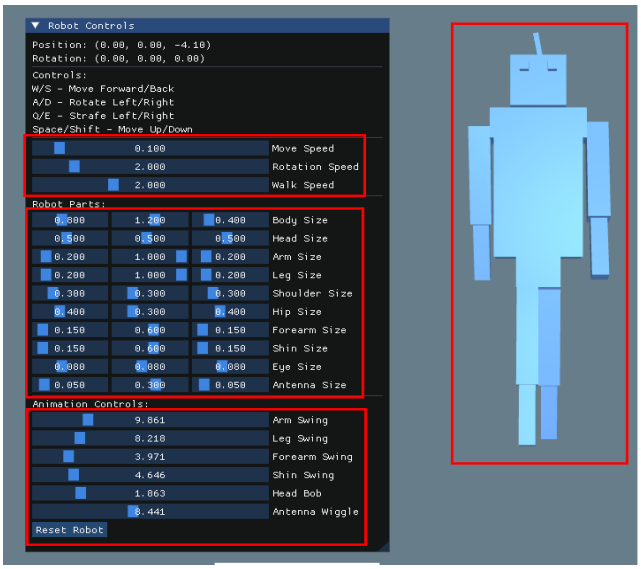
Система за управление

- **Обработка на потребителски вход**

```
void key_callback(GLFWwindow* window, int key, int action, int mods) {
    if (action == GLFW_PRESS || action == GLFW_REPEAT) {
```

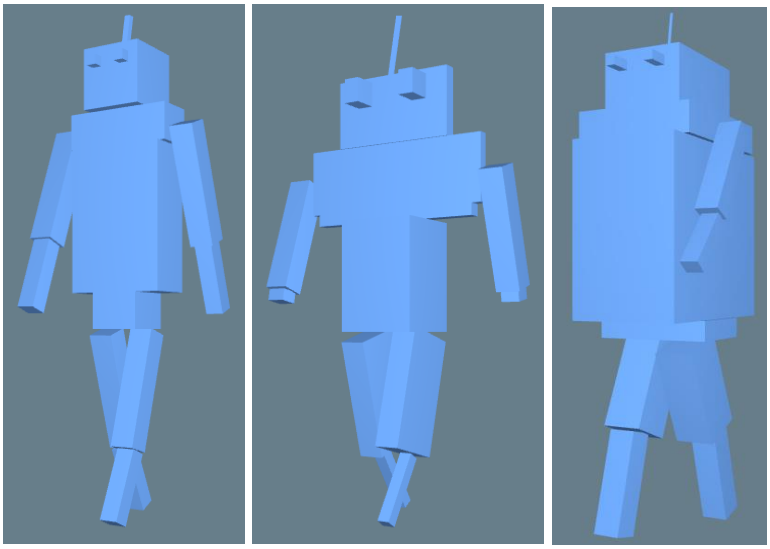
```
        switch (key) {
            case GLFW_KEY_W:
                cg::robot.position.x -=
sin(glm::radians(cg::robot.rotation.y)) * g_move_speed;
                cg::robot.position.z -=
cos(glm::radians(cg::robot.rotation.y)) * g_move_speed;
                break;
        }
    }
}
```

Преглед на интерфейса

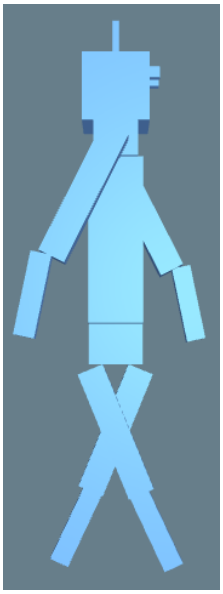


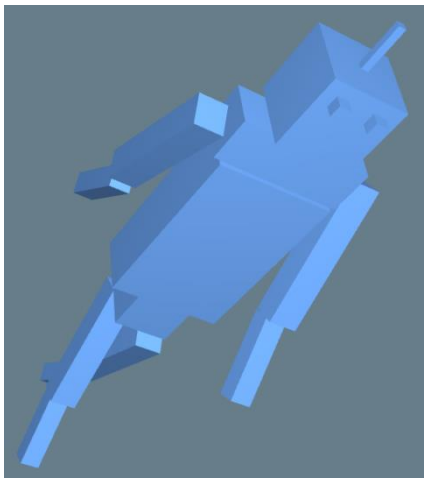
Фиг №1

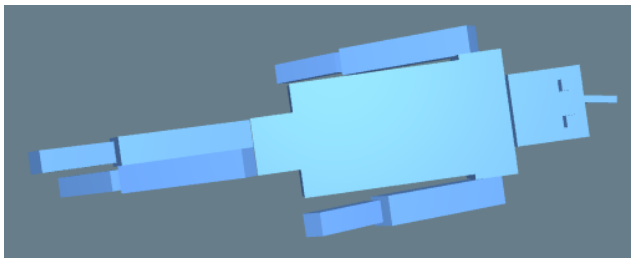
При стартиране на проекта се изчертава робота, който се намира в дясната част на фиг №1. От лявата страна е разположено меню, чрез което потребителят може да променя различни атрибути по робота – промяна на дължината, широчината и дебелината на части от тялото на робота. Освен това, може да се промени и скоростта на движение на крайниците на робота, както е скоростта на ротиране и транслиране на обекта. Най-отдолу има бутон за възстановяване на първоначалния изглед на робота (reset на сцената и атрибутите).



Тук са представени различни изгледи на робота след промяна на атрибутите му. Първият робот е първоначално заредения (дефолтен) робот, докато останалите два са потребителски генерирани след използване на менюто за промяна. Тук ясно е показано какви промениу по изгледа на робота са възможни – дължина, ширина и дебелина в зависимост от осите, по които се прави промяната.







Тук се демонстрират част от възможностите за ротация по осите x, y, z. Освен това робота може да се скалира (мащабира)

Потребителското меню за управление на робота. Показва информация за текущата позиция на обекта, контролите, с които се управлява и слайдери за динамична промяна на атрибутите. Накрая има и бутони за възстановяване на първоначалното състояние на робота.

Заключение

Проектът успешно демонстрира:

- Ефективна 3D визуализация с използване на съвременни графични техники
- Сложна йерархична анимация на многокомпонентен обект
- Интуитивно управление както чрез клавиатура, така и чрез графичен интерфейс
- Модулна архитектура, позволяваща лесно разширяване

Пропуски и ограничения

- Липса на усъвършенствано осветление - може да се добавят по-сложни модели на осветление
- Ограничени възможности за анимация - липсват преходи между анимационни състояния
- Проста геометрия - използва се само кубична геометрия

Литература

1. Angel, E., & Shreiner, D. (2014). *Interactive Computer Graphics: A Top-Down Approach with WebGL*. Pearson Education.
2. Hughes, J. F., et al. (2013). *Computer Graphics: Principles and Practice*. Addison-Wesley Professional.
3. Dunn, F., & Parberry, I. (2011). *3D Math Primer for Graphics and Game Development*. A K Peters/CRC Press.
4. Schneider, P. J., & Eberly, D. H. (2003). *Geometric Tools for Computer Graphics*. Morgan Kaufmann.
5. Parent, R. (2012). *Computer Animation: Algorithms and Techniques*. Morgan Kaufmann.
6. Kessenich, J., et al. (2016). *OpenGL Programming Guide*. Addison-Wesley Professional.
7. Rost, R. J., & Licea-Kane, B. (2009). *OpenGL Shading Language*. Addison-Wesley.
8. Wright, R. S., et al. (2018). *OpenGL SuperBible*. Addison-Wesley.
9. Shneiderman, B., et al. (2016). *Designing the User Interface*. Pearson Education.
10. Akenine-Möller, T., et al. (2018). *Real-Time Rendering*. A K Peters/CRC Press.

Допълнителни източници

https://www.youtube.com/playlist?list=PLlrATfBNZ98foTJPJ_Ev03o2oq3-GGOS2

https://www.youtube.com/c/TheChernoProject

https://www.youtube.com/watch?v=nlKc5U2k9kY

https://www.youtube.com/c/Computerphile/search?query=graphics

https://www.youtube.com/playlist?list=PLZHQObOWTQDPD3MizzM2xVFitgF8hE_ab

https://www.youtube.com/watch?v=O7WenN8KhQk