**Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»**

**кафедра ПМ иК**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Структуры и алгоритмы обработки данных»**

**Вариант №74**

**Выполнил:студент группы ИП-913**

**Степин Александр Вячеславович**

**Проверил: доцент кафедры ПМиК**

### **Суходоева Наталья Николоаевна**

**Новосибирск – 2020**

**ОГЛАВЛЕНИЕ**

**1.** [**Поста**](#_Toc515570163)**новка задачи (вариант №74)**

**2. Основные идеи и хаpактеpистики пpименяемых методов**

**Метод сортировки**

**Двоичный поиск**

**Списки и очереди**

**Вид дерева и поиск**

**Метод кодирования**

**3. Описание структур данных и использованных алгоритмов**

**Использованные структуры данных**

**Особенности реализации алгоритмов**

**4. Описание программы**

**Основные переменные и структуры**

**Описание подпрограмм (название, параметры, что делает)**

**Списки и очереди**

**Вид дерева и поиск**

**Метод кодирования**

**5. Исходный текст пpогpаммы (выделить заголовки подпрограмм)**

**6. Результаты (показательные фрагменты, 3-5 страниц)**

**7. Выводы (как решена поставленная задача)**

[**Поста**](#_Toc515570163)**новка задачи (вариант №74)**

Т. к. формулы движения по окружности подразумевают точку, а не объект с заданным размером, то сведем задачу к работе с точкой. Но для начала определим размер и форму фигуры. Фигура составная, следовательно, строится путем комбинации простейших геометрических фигур. В этом варианте задания реализовано нечто похожее на космический корабль, как показано в примере, и оно двигается и вращается. Для двигателей используются прямогульники, для “”тела” корабля – круг, для “носа” – треугольник. Для удобства при проверке выхода за границы видимой области обозначим форму фигуры как прямоугольник наименьшей площади, такой, что в него помещается вся наша составная фигура. Обозначим точку, которая находится на пересечении левой и верхней сторон нашего прямогульника. Она и будет двигаться по окружности.

**Основные идеи и хаpактеpистики пpименяемых методов**

***a. М****етод сортировки*

Учитывая иерархию классов, объектов и их методов, алгоритм работы программы следующий:

***a. Д****воичный поиск*

Учитывая иерархию классов, объектов и их методов, алгоритм работы программы следующий:

***a. М****етод сортировки*

Учитывая иерархию классов, объектов и их методов, алгоритм работы программы следующий:

***a. М****етод сортировки*

Учитывая иерархию классов, объектов и их методов, алгоритм работы программы следующий:

**Описание алгоритма программы**

***a.*** *Алгоритм программы*

Учитывая иерархию классов, объектов и их методов, алгоритм работы программы следующий:

Приложение принимает на вход простейшие геометрические фигуры, которые объединяет в одну составную, вычисляет размер прямоугольной области, такой, что её площадь наименьшая и каждая из простейших фигур находится в этой области и не выходит за её границы.

За объект, движущийся по окружности, принимается точка, находящаяся в вехнем левом углу получанной прямоугльной области. За объект вращающийся и находящийся паралельно касательной к окружности принимается верхняя сторона пряоугльной области.

Программа вращает и передвигает фигуру в круговом движении до тех пор, пока не будет закрыта.

***b.*** *Выводы по разделу #2*

Так как каждый описанный выше этап работы программы имеет вполне определённое поведение при различных входных данных (за исключением багов, недочётов и уязвимостей которые будут выявляться в ходе эксплуатации программы при условии поддержки и развития проиложенния разработчиком, то есть мной (-: ), можем считать процесс работы данной программы алгоритмом.

**Вывод**

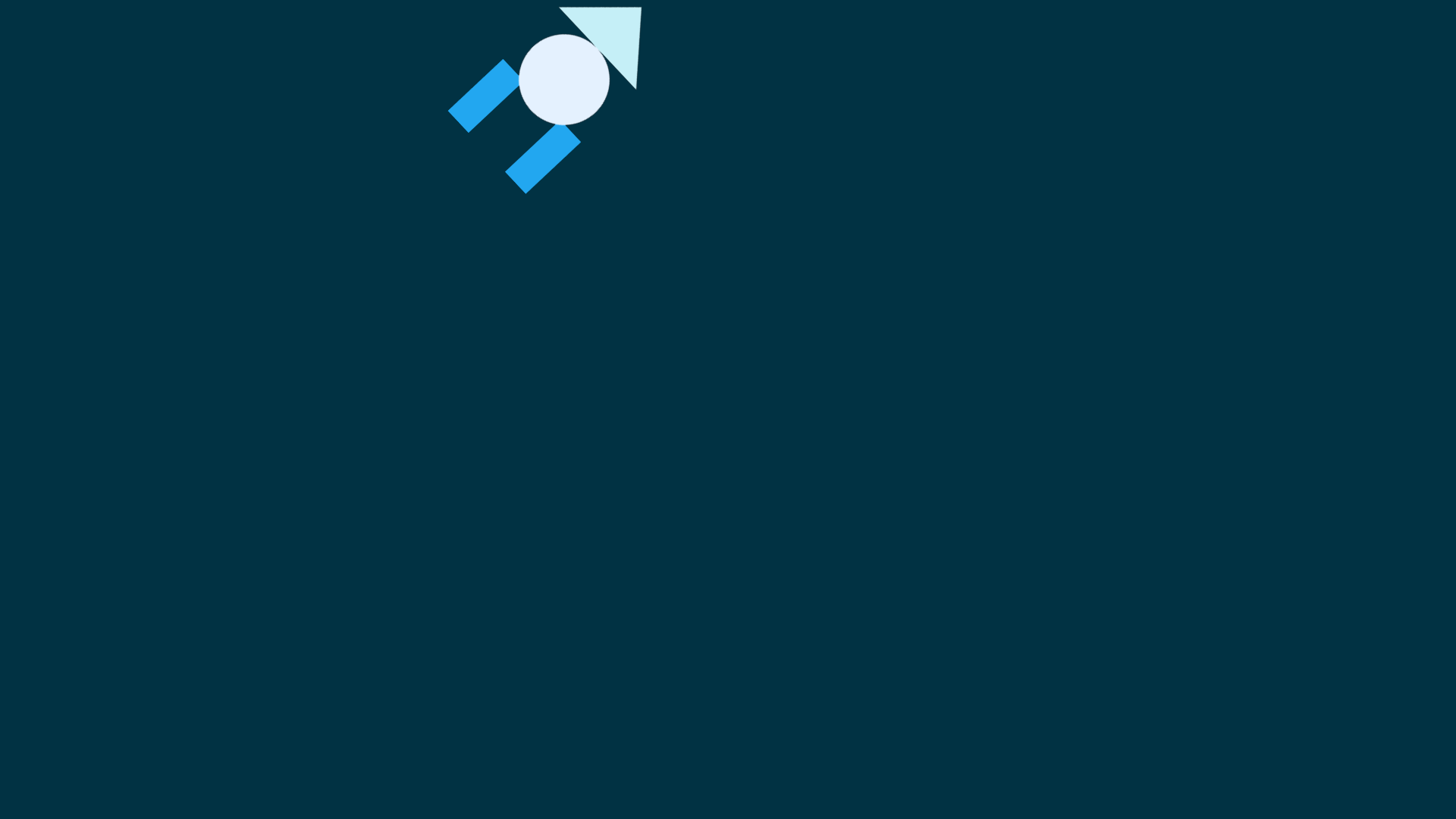
Для реализации программы были задействованы математические формулы, ООП и другие возможности, предоставляемые языком c++. Использованы наследование, перегрузка, ассоциация (агрегация), виртуальные методы, инкапсуляция и др. концепции и технологии. ООП позволило создать хорошую архитектуру приложения и предоставить необходимый интерфейс для взаимодействия. Были использованы формулы движения по окружности, реализовано виртуальное окно как интерфейс для работы с составной фигурой используя точку для перемещения. ООП показало эффективность в конексте данной программы.

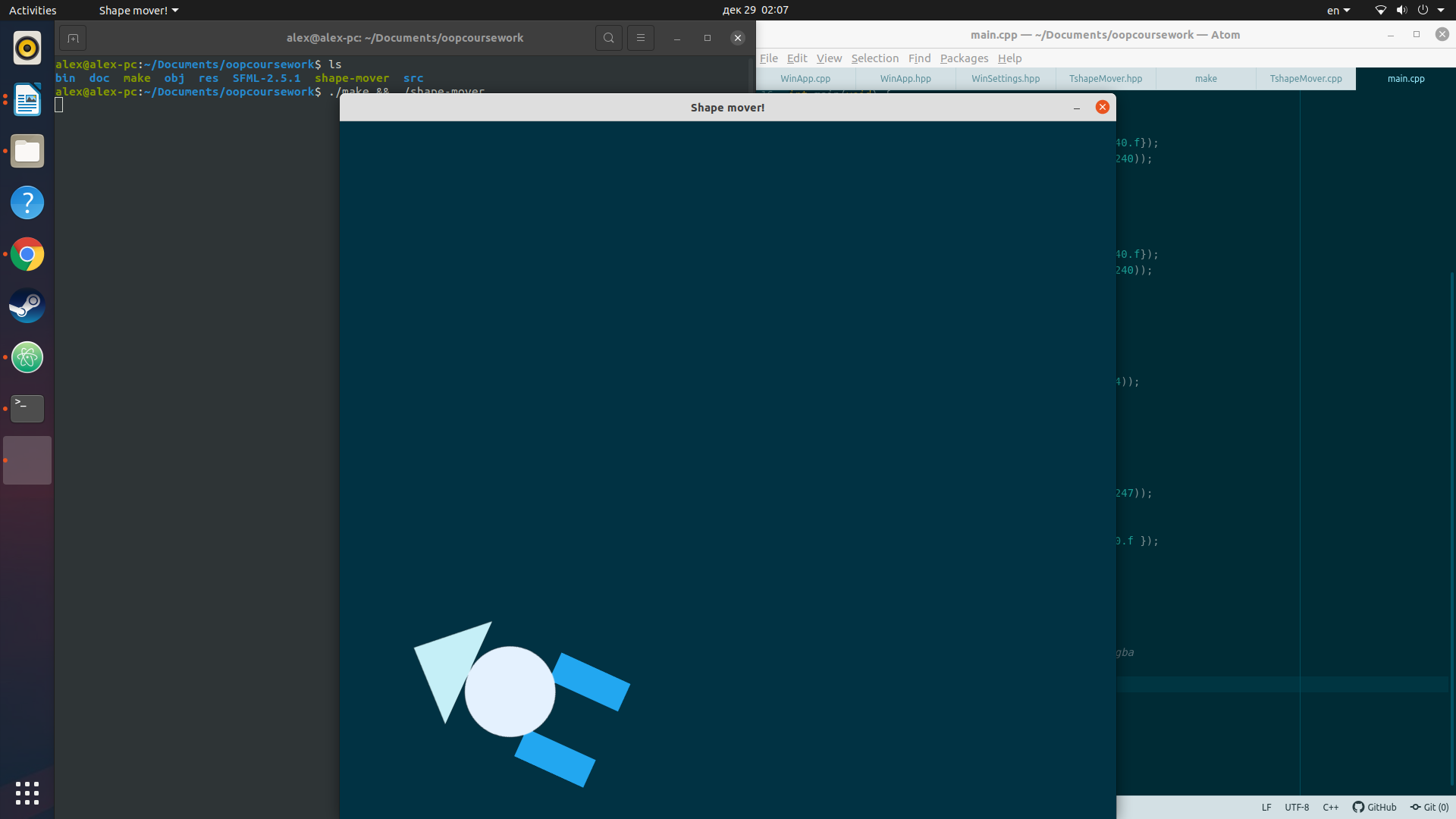
**Скриншоты работы программы**

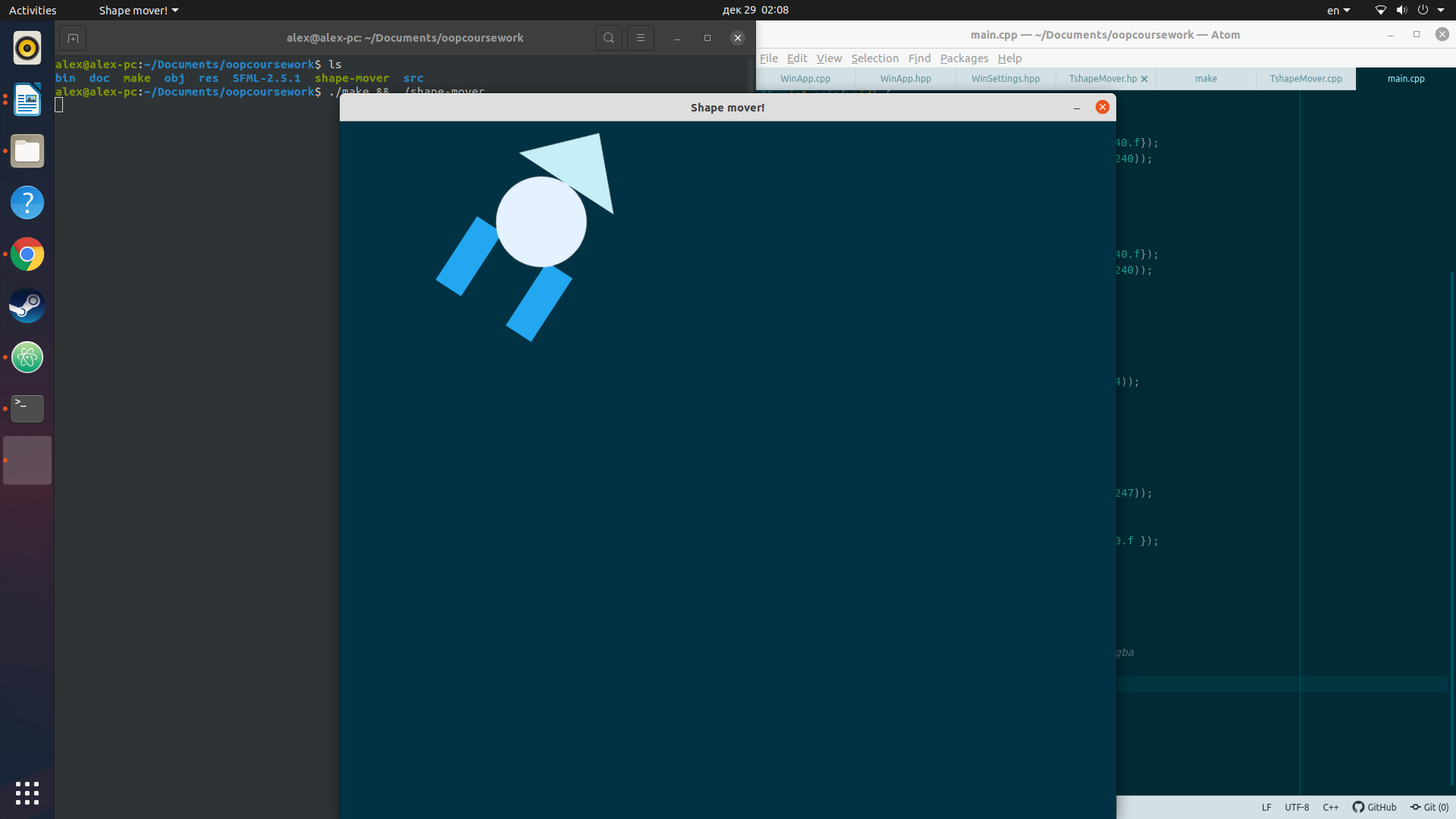
***a.*** *Полноэкранные скриншоты*

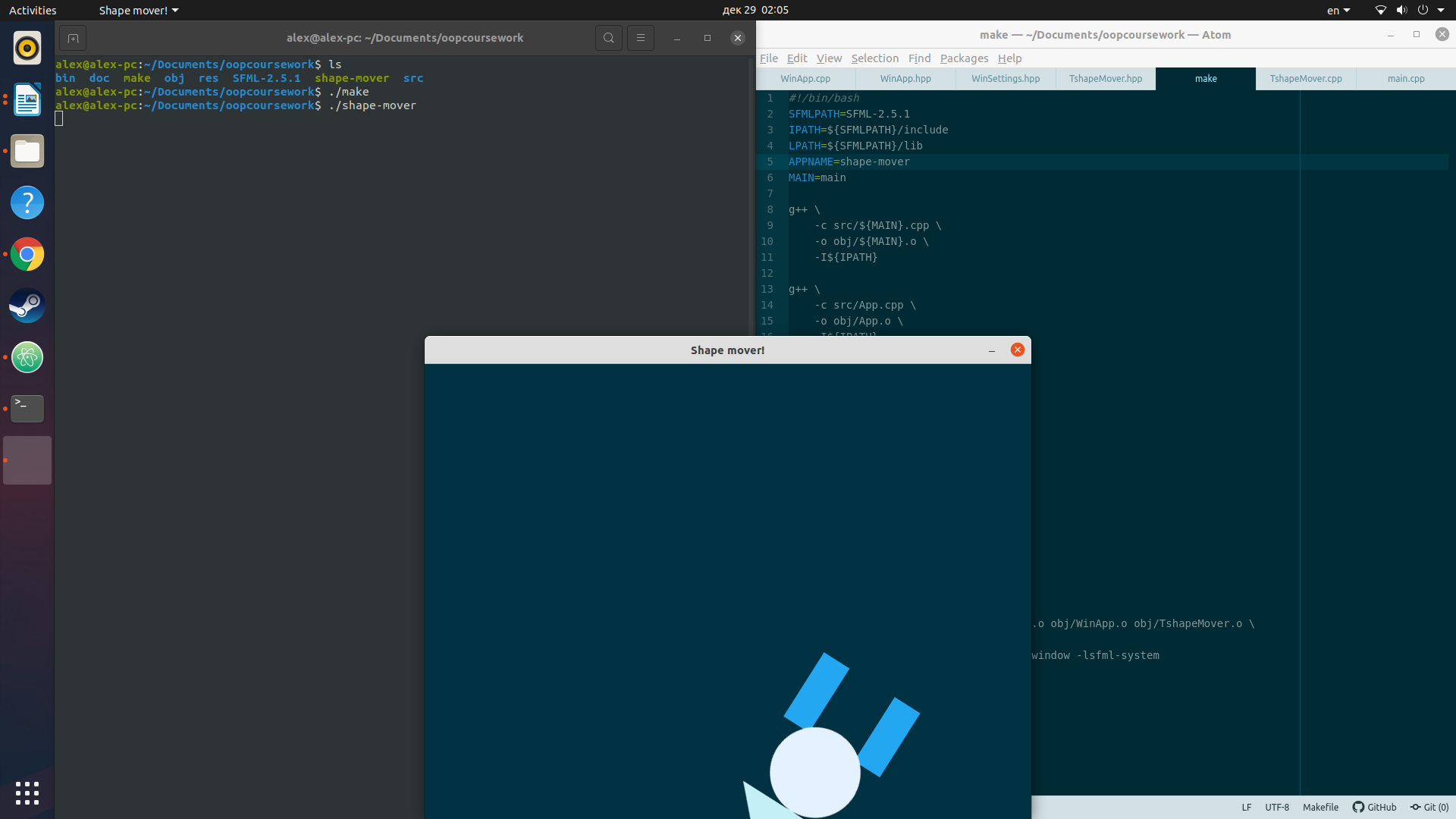
Скриншот 1: Полноэкранный режим. Круговое вращение составного объекта

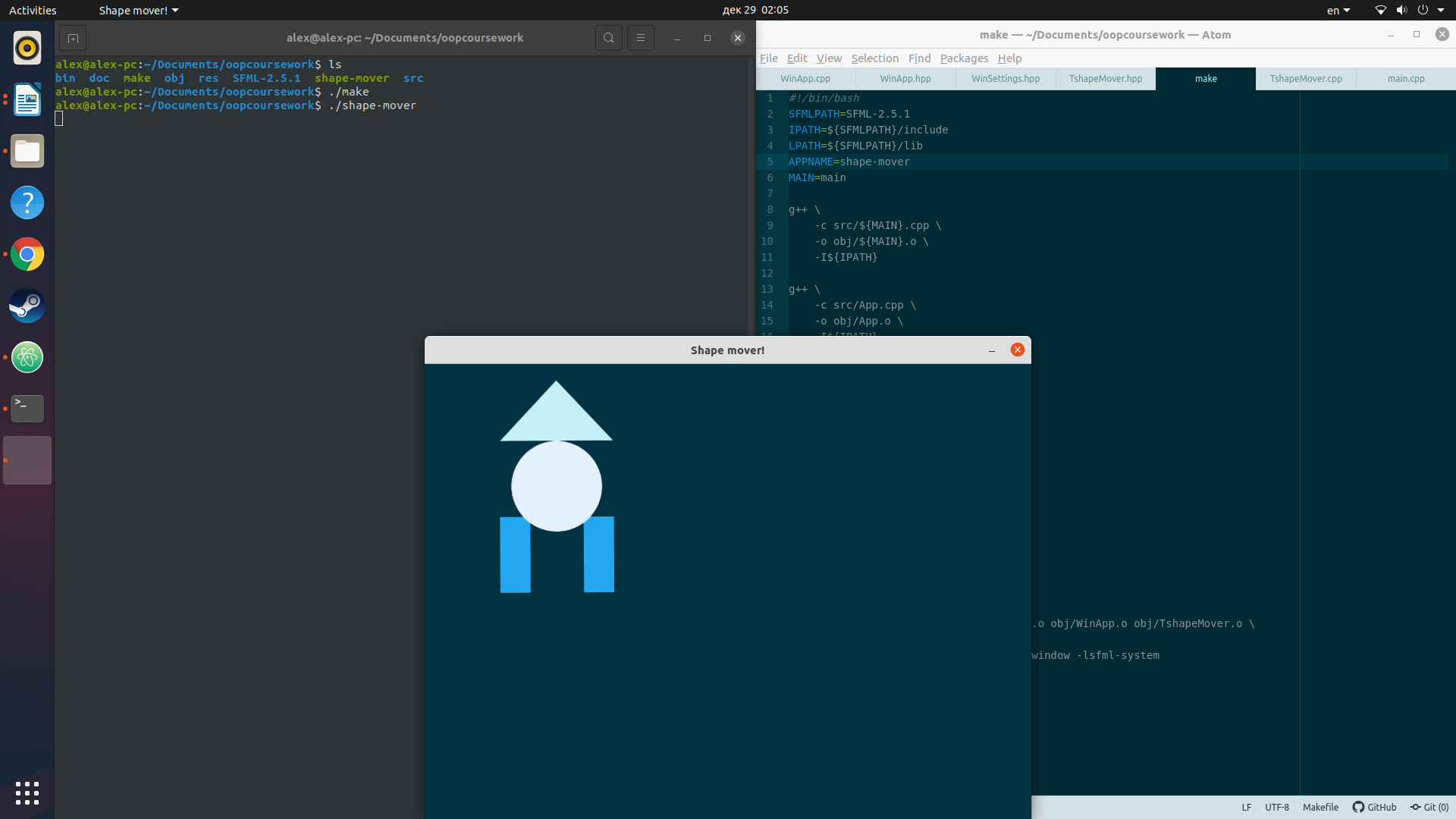
Скриншот 2: Полноэкранный режим. Круговое вращение составного объекта

Скриншот 3: Полноэкранный режим. Круговое вращение составного объекта

Скриншот 4: Оконный режим. Круговое вращение составного объекта

Скриншот 5: Оконный режим. Круговое вращение составного объекта

Скриншот 6: Оконный режим. Круговое вращение составного объекта

Скриншот 7: Оконный режим. Круговое вращение составного объекта

**Листинг кода**

Подсветка синтаксиса реализована с помощью онлайн сервиса подсветки кода. С нумерацией строк посердством функций сервиса ломаются отступы (полностью исчезают). С нумерацией посредством функций LibreOffice лтступы тоже ломаются! Они есть, но они сломаны. Поэтому первый файл будет с нумерацией, остальные могут быть без.

***b.*** *App.hpp*

#pragma once #include <string>

class App {

public:

std::string appName;

void setAppName(std::string name);

App() = delete;

protected:

App(std::string name);

public:

virtual void run() = 0;

std::string getAppName();

};

***c.*** *WinApp.hpp*

#pragma once

#include <string>

#include <SFML/System/Vector2.hpp>

#include <SFML/Window/ContextSettings.hpp>

#include <SFML/Window/VideoMode.hpp>

#include <SFML/Graphics/RenderWindow.hpp>

#include <SFML/Window/WindowStyle.hpp>

#include "App.hpp"

#include "WinSettings.hpp"

class WinApp : public App {

private:

WinApp() = delete;

protected:

WinSettings windowSettings;

sf::RenderWindow window;

sf::VideoMode getVideoMode();

unsigned getWindowStyle();

bool isWindowed();

void setSettings(WinSettings);

void render();

sf::Vector2u getSize();

WinApp(std::string name, WinSettings);

WinApp(std::string name);

};

***d.***  *WinSettings.hpp*

#pragma once

#include <SFML/Window/ContextSettings.hpp>

#include <SFML/Window/VideoMode.hpp>

#include <SFML/Graphics/RenderWindow.hpp>

#include <SFML/Window/WindowStyle.hpp>

struct WinSettings {

public:

sf::VideoMode videoMode = sf::VideoMode::getDesktopMode();

sf::ContextSettings contextSettings = sf::ContextSettings(0, 0, 4);

unsigned windowStyle = sf::Style::Fullscreen;

sf::Color backgroundColor = sf::Color::Black;

};

***e.***  *TshapeMover.hpp*

#pragma once

#include "WinApp.hpp"

#include "WinSettings.hpp"

#include "Tshape.hpp"

#include <cmath>

#include <SFML/System/Vector2.hpp>

#include <SFML/Window/Keyboard.hpp>

#include <SFML/Window/ContextSettings.hpp>

#include <SFML/Window/Event.hpp>

#include <SFML/Graphics/Color.hpp>

//

// ::TshapeMover

class TshapeMover : public WinApp {

public:

class Subwindow;

struct Settings;

private:

Tshape\* shape;

// bool isMoving = true;

TshapeMover();

Settings\* settings;

void setupSettings();

void setupSettings(Settings\*);

public:

void handleKeyboardEvent(sf::Event\* event);

void handleEvent(sf::Event\* event);

void handleSettings(Settings\*);

void checkEvents();

bool redraw(Subwindow\* subwindow, float t);

void run() override;

void shutDown(void);

TshapeMover(Settings\* settings, Tshape\* shape);

};

//

// ::TshapeMover::Settings

struct TshapeMover::Settings {

public:

sf::Color backgroundColor = sf::Color::Black;

sf::Vector2u windowSize = {800, 600};

bool isWindowed = false;

};

//

// ::TshapeMover::Subwindow

class TshapeMover::Subwindow {

private:

sf::Vector2u subwindowSize;

sf::Vector2u windowSize;

sf::Vector2u translation = {0, 0};

public:

Subwindow(sf::Vector2u windowSize);

bool translate(sf::Vector2u translation);

sf::Vector2f getPosition(sf::Vector2f coords);

bool isOutside(sf::Vector2f coords);

bool setSize(sf::Vector2u size);

sf::Vector2u getSize(void);

};

***f.***  *Tshape.hpp*

#pragma once

#include <vector>

#include <utility>

#include <functional>

#include <SFML/System/Vector2.hpp>

#include <SFML/Window/ContextSettings.hpp>

#include <SFML/Graphics/RenderWindow.hpp>

#include <SFML/Graphics/RenderTexture.hpp>

#include <SFML/Graphics/Sprite.hpp>

#include <SFML/Graphics/Shape.hpp>

class Tshape {

using TcreateShape = std::function <sf::Shape\*()>;

using BasicShape = sf::Shape;

private:

sf::RenderTexture texture;

sf::Sprite sprite;

sf::Vector2u size = {0, 0};

std::vector<BasicShape\*> basicShapes;

sf::Vector2u computeSize();

public:

sf::Vector2u getSize();

void draw(sf::RenderWindow\* w);

void setPosition(float x, float y);

void setRotation(float angle);

void build(sf::ContextSettings);

Tshape(std::vector<TcreateShape>);

~Tshape();

};