Постановка задачі

Скласти програму, яка в діалоговому режимі керує графічними об'єктами, що відображаються на екрані дисплея.

Програма повинна підтримувати такі загальні елементи поведінки графічних об'єктів:

1. Активізація/візуалізація за вибором.
2. Переміщення зі слідом/без.
3. Відновлення початкового стану образа.
4. Зміна кольору.
5. Зміна стану видимий/невидимий.
6. При агрегації об'єктів забезпечити можливість агрегації агрегатів.

Поведімка об'єктів:

1. Завершення роботи з об'єктом - зникає.
2. Рух об'єкта в автоматичному режимі - по заданому закону.
3. Зміна кольору - під впливом іншого об'єкта.
4. Деформація образа - по команді
5. Зборка/запам'ятання агрегованого образа - видаленням.

Опис розв'язку

Для розв'язку задачі було створено базовий клас геометричної фігури - Shape. В ньому реалізовані всі необхідні базові властивості:

- float getX() const; float getY() const; void setLocation(float x, float y); - робота з положенням фігури в просторі
- float getWidth() const; float getHeight() const; void setSize(float h, float w); - робота з розміром фігури
- virtual float getMinX() const; virtual float getMinY() const; virtual float getMaxX() const; virtual float getMaxY() const; - робота з границями фігури
- bool isVisible() const; void setVisible(bool visible); - робота з видимістю фігури
- float getRed() const; float getBlue() const; float getGreen() const; void setColor(float r, float g, float b); - отримання та встановлення компонент кольору
- virtual void draw(); - відповідає за візуалізацію фігури
- virtual void setScale(float scalef); - масштабує фігуру
- Shape(); Shape(float x, float y, float h, float w, float r, float g, float b); - конструктори
- Shape(const Shape& shape); - конструктор копіювання
- Shape& operator=(const Shape& shape); - оператор присвоювання
- bool operator==(const Shape& shape) const; - оператор порівняння

Для використання в програмі було реалізовані 2 геометричні об'єкти - коло(Circle) та прямокутник(Rectangle), що які являються наслідками класу фігури(Shape).

зміни в класі Circle:

- float getRadius() const; void setRadius(float radius); - повертає та встановлює радіус кола
- Circle(float x, float y, float radius, float r, float g, float b); - конструктор з можливістю задати радіус
- virtual void draw(); - перевантажена функція, в ній реалізовано візуалізація кола

зміни в класі Rectangle:

- virtual void draw(); - перевантажена функція, в ній реалізовано візуалізація прямокутника

Також було реалізовано клас, що являється контейнером для фігур, наслідків класу Shape. Він також е наслідком класу Shape:

- bool isSelected() const; void setSelected(bool selected); - повертає та вставновлює, чи є дана фігура вибраною
- std::vector<Shape*> getShapes(); - вектор фігур, які знаходяться в данному контейнері
- void add(Shape* shape); - додає фігуру до контейнеру
- void remove(Shape* shape); - видаляє фігуру з контейнеру
- void eraseAll(); - видаляє всі фігури з контейнеру
- virtual void draw(); - перевантажена функція візуалізації. Якщо контейнер вибраний, то візуалізується границя. Потім виклакається функція візуалізаціїї у всіх фігур, о містяться в контейнері
- void move(float dx, float dy); - зміює положення контейнеру на dx та dy по вісям Ox та Oy

Для візуалізації використовується бібліотека OpenGL. Для відображення вікна з об'єктами та реагування на дії користувача(зміна розміру вікна, натискання клавіш на клавіатурі та ін) використовується бібліотека GLUT (openGL Utility Toolkit).

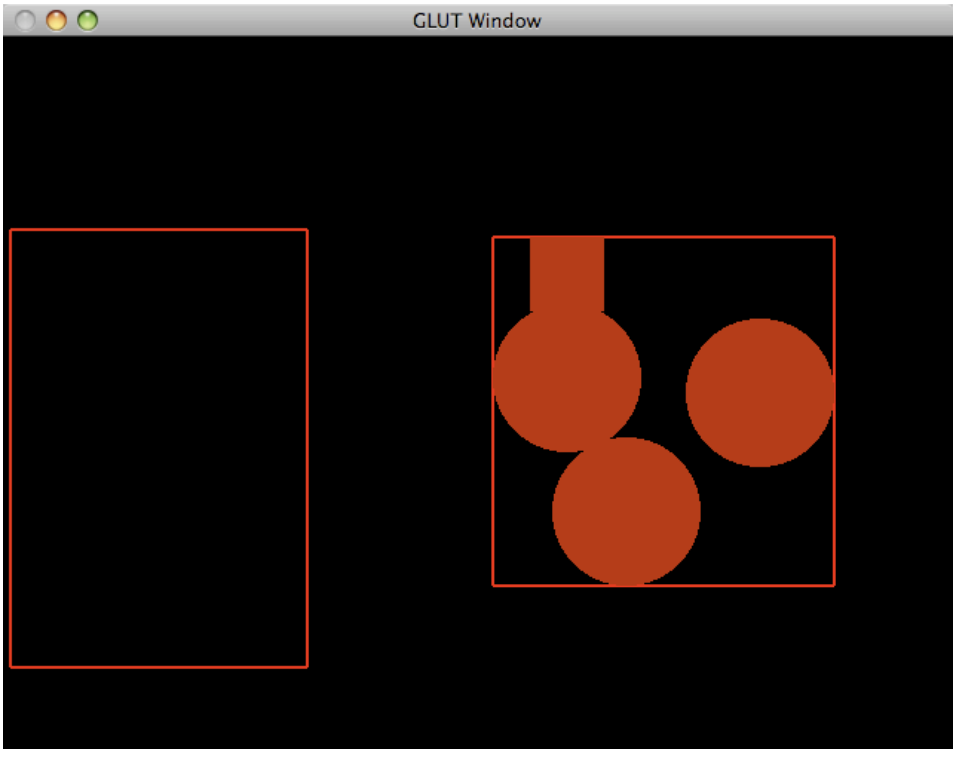
Вихідний текст програми розв'язку задачі

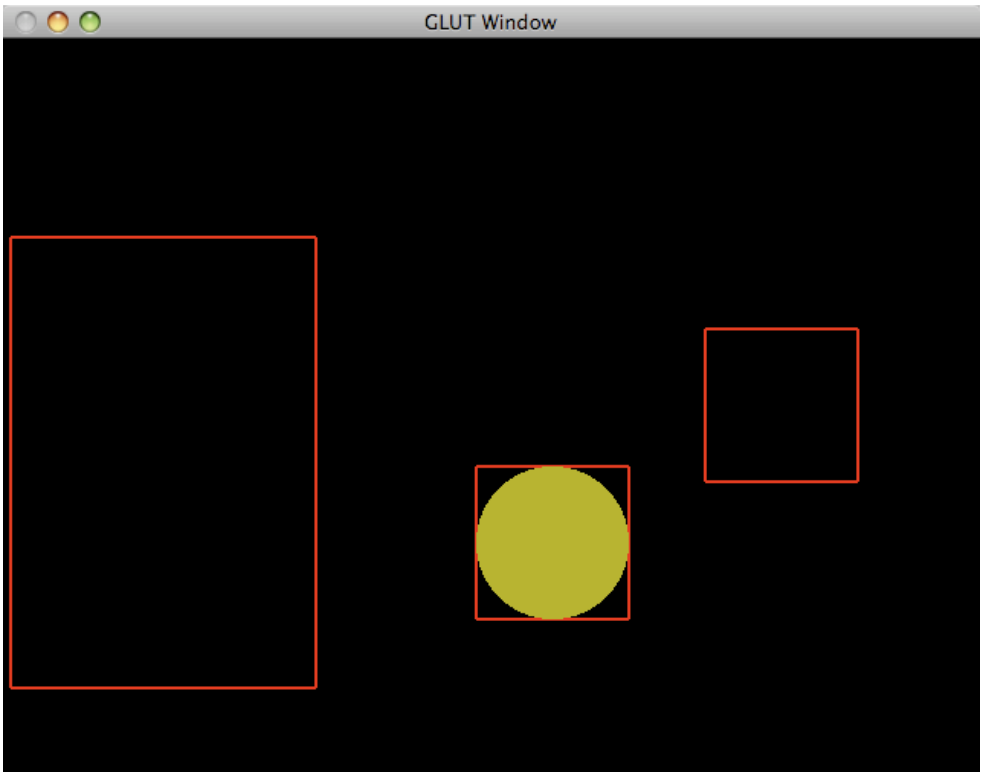
див. Додаток 1

Опис інтерфейсу (керівництво користувача)

Коло додається до сцени за кнопкою 1, прямокутник - 2. Перехід між фігурами здійснюється за клавішею Tab. Переміщення фігур на площині здійснюється за допомогою W(вгору), S(вниз), A(вліво), D(вправо). Агрегація фігур здійснюється за кнопкою M. Масштабування - за Z(збільшення), X(зменшення).

Опис тестових прикладів





результати роботи повністю відповідають постановці задачі.