Семинар #2: Наследование. GUI. Домашнее задание.

Наследование

Задача 1. Фигуры

В файле `shape.cpp` написаны простейшие классы `Circle`, `Rectangle` и `Triangle`, описывающие геометрические фигуры: круг, прямоугольник и треугольник. При написании этих классов наследование не было использовано, в следствии чего некоторые поля и методы одинаковы у всех классов (поле `mPosition` и методы `getPosition` и `setPosition`).

- Создайте класс `Shape`, который бы описывал абстрактную фигуру и содержал бы поля и методы, общие для всех фигур. Измените код классов `Circle`, `Rectangle` и `Triangle` так, чтобы они наследовались от класса `Shape`. Общие для всех фигур поля и методы должны содержаться только в классе `Shape`, но не в классах-наследниках.
- Добавьте метод `void move(Vector2f change)` в класс `Shape`. Этот метод должен изменять поле `mPosition` на значение `change`. Так как остальные классы наследуются от `Shape`, то этот метод можно будет вызвать у всех объектов дочерних классов. Протестируйте этот метод, изменив положения объектов дочерних классов.

Задача 2. Приведение типов при наследовании

Пусть есть два следующих класса:

```
struct Alice
{
    int x;
    void func() const {std::cout << "Alice " << x << std::endl;}
};

struct Bob : public Alice
{
    int y;
    void func() const {std::cout << "Bob " << x << " " << y << std::endl;}
};
```

Определите, скомпилируется ли следующий код, использующий эти классы, и, если скомпилируется, то что будет напечатано в следующих программах:

1. ```
   Alice a {10};
   Bob b {20, 30};
   a = b;
   a.func();
   ```
2. ```
   Alice a {10};
   Bob b {20, 30};
   b = a;
   b.func();
   ```
3. ```
   Bob b {20, 30};
   Alice* p = &b;
   p->func();
   ```
4. ```
   Alice a {10};
   Bob* p = &a;
   p->func();
   ```

Пусть есть следующие классы:

```
struct Alice
{
    int x;
    void func() const {std::cout << "Alice " << x << std::endl;}
};

struct Bob : public Alice
{
    int y;
    void func() const {std::cout << "Bob " << x << " " << y << std::endl;}
};

struct Casper : public Bob
{
    int z;
    void func() const {std::cout << "Casper " << x << " " << y << " " << z << std::endl;}
};
```

Определите, скомпилируется ли следующий код, использующий эти классы, и, если скомпилируется, то что будет напечатано в следующих программах:

5. ```
   Alice a {10};
   Bob b {20, 30};
   Casper c {40, 50, 60};
   a = c;
   a.func();
   ```

6. ```
   Alice a {10};
   Bob b {20, 30};
   Casper c {40, 50, 60};
   Alice* p = &c;
   p->func();
   ```

7. ```
   Alice a {10};
   Bob b {20, 30};
   Casper c {40, 50, 60};
   Bob* p = &a;
   p->func();
   ```

8. ```
   Alice a {10};
   Bob b {20, 30};
   Casper c {40, 50, 60};
   Bob* p = &c;
   p->func();
   ```

Для того, чтобы сдать эту задачу нужно создать файл в формате `.txt` и, используя любой текстовый редактор, записать в него ответы в следующем формате (ответы ниже неверны):

```
1) Error
2) Alice 10
3) Bob 20 30
...
```

После этого, файл нужно поместить в ваш репозиторий на github.

Задача 3. Изменение цвета

В файле `draggable.cpp` написан класс `Draggable`, который описывает передвигаемый курсором мыши прямоугольник. Ваша задача – написать класс `DraggableWithColorChange` – наследник класса `Draggable`. Новый класс должен также описывать передвигаемый прямоугольник, но, во время передвижения прямоугольника, его цвет должен меняться на другой. Конструктор нового класса будет иметь вид:

```
DraggableWithColorChange(Vector2 position, Vector2 size, Color baseColor, Color dragColor)
```

Где `baseColor` – это основной цвет прямоугольника, а `dragColor` – цвет прямоугольника при перетаскивании. Протестируйте этот класс в функции `main`.

Задача 4. Окна

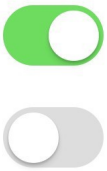
В файле `windows.cpp` написан класс `BaseWindow`, описывающий простейшее окно. Это окно состоит из двух прямоугольников. Первый прямоугольник опредяляет границы отрисовки окна, а второй опредяляет границы области, за которую прямоугольник можно перетаскивать.

- Создай класс `MessageWindow`, наследник `BaseWindow`. У объектов этого класса, помимо функционала `BaseWindow` должен быть текст, в котором отображается некоторая строка.
- Создайте классы `ErrorWindow` и `DoneWindow`, наследники класса `MessageWindow`. Эти два окна должны отличаться от окна `MessageWindow` только тем, что окно типа Error должно всегда рисоваться оттенком красного цвета, а окно типа Done - оттенком зелёного цвета.
- Создай класс `QuestionWindow`, наследник `BaseWindow`. У объектов этого класса, помимо функционала `BaseWindow` должен быть текст, в котором отображается некоторая строка. А также две кнопки внизу: Ok и Cancel. При нажатии на кнопку Ok в консоль должно выводиться строка `Ok`, а при нажатии на кнопку Cancel – строка `Cancel`. Используйте класс `Button` из файла `button.hpp` для создания кнопок.

Создание классов графического интерфейса

Задача 5. Тумблер

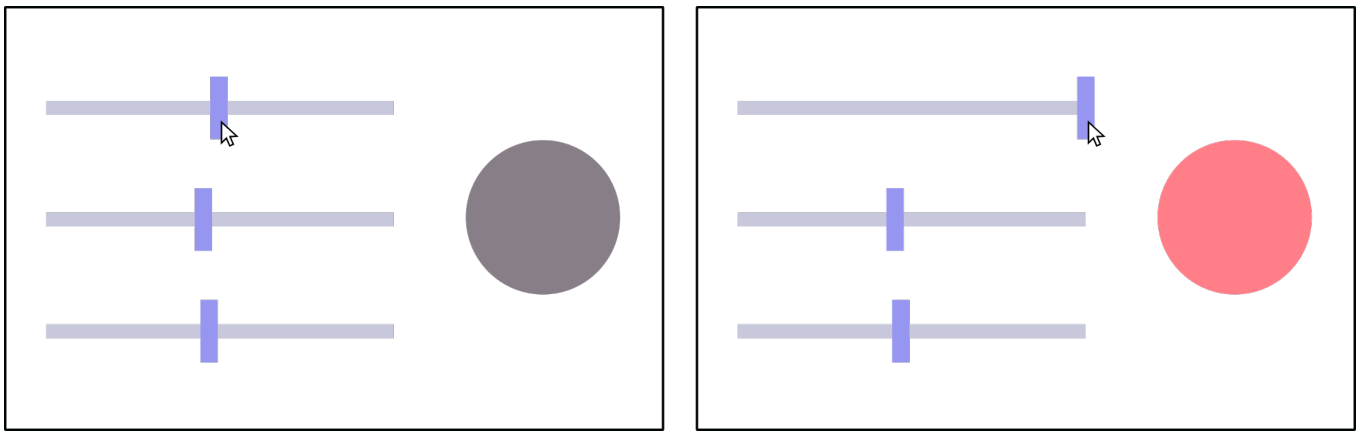
Создайте класс `Toggle`, который бы релизовывал элемент интерфейса тумблер (переключатель).



После того, как создадите класс, напишите программу, в которой бы создавалось 10 тумблеров. Все тумблеры должны работать. Программа должна корректно работать при изменении размеров окна.

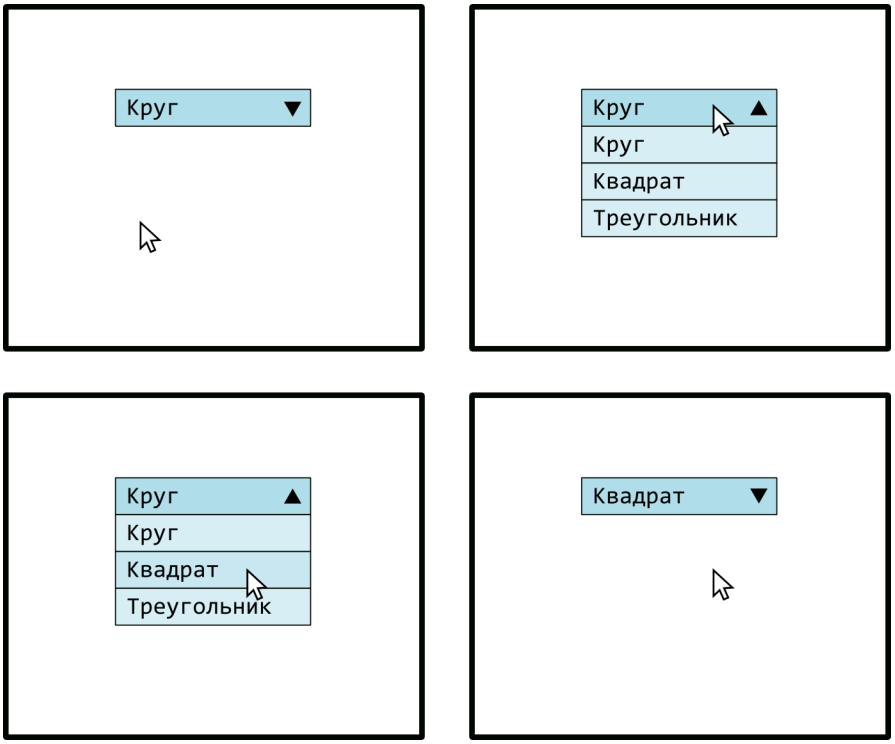
Задача 6. Три слайдера

Напишите программу, которая создаёт круг и три слайдера. Цвет круга должен меняться при изменении положения ползунков слайдеров. Каждый слайдер должен отвечать за свою компоненту цвета круга (красную, зелёную или синию). Используйте класс `Slider` из `classroom_tasks/code/5gui/2slider`.



Задача 7. Выпадающий список

Создайте класс `DropList`, который бы релизовывал элемент интерфейса выпадающий список без полосы прокрутки.



После создания класса напишите программу, которая реализует выпадающий список с тремя вариантами: `Круг`, `Квадрат` и `Треугольник`. В зависимости от выбранного в списке варианта, на экране должна отображаться соответствующая фигура: круг, квадрат или треугольник.

Задача 8. Select Move Delete

В папке `select_move_delete/` содержится заготовка исходного кода для этого задания. В этой программе есть несколько объектов(кругов), которые можно выделять. Выделение происходит по нажатию левой клавиши мыши. Зажав клавишу `Ctrl` можно выделить несколько объектов. Также в программе реализован прямоугольник выделения (но он пока не выбирает объекты). Левой кнопкой мыши с зажатой клавишей левый `Alt` можно создать круг случайного размера. Добавьте следующие возможности в программу:

- Задание случайного цвета. При нажатии клавиши пробел цвет всех выделенных шаров должен меняться на случайный. Для этого понадобится добавить поле `color` в класс `Ball`.

- Выделение объектов с помощью прямоугольника выделения. Прямоугольник выделения должен рисоваться только если нажатие мыши произошло вне кругов. Все объекты, полностью находящиеся внутри прямоугольника выделения, на момент отпускания левой кнопки мыши должны выделяться. Объекты должны выделяться во время изменения прямоугольника, а не только при отпускании кнопки мыши.

- Перемещение всех выделенных объектов при зажатии левой клавиши мыши и её движении. Перемещаться должны все выделенные объекты параллельно (также как перемещаются несколько выделенных значков на рабочем столе). Прямоугольник выделения при этом рисоваться не должен.

- При нажатии клавиши `Delete`, все выделенные объекты должны удаляться. Чтобы удалить элемент из `std::list` используйте итераторы и метод `erase`. При удалении элемента `std::list` нужно внимательно следить за тем, чтобы не испортить итераторы.

- Контекстное меню должно открываться при нажатии правой кнопки мыши. Добавьте следующие варианты в меню:

  - `Delete` - при нажатии на выделенные объект правой кнопкой мыши и выборе этого варианта все выделенные объекты должны удаляться.
  - `Create` - при нажатии на любое место и выборе этого варианта должнен создаваться новый случайный круг.
  - `Random Color` – все выделенные объекты должны окрашиваться в случайный цвет.
  - `Increase` – все выделенные объекты должны увеличиваться на 25 процентов.
  - `Decrease` – все выделенные объекты должны уменьшаться на 25 процентов.

- **Copy Paste Cut** Добавьте возможность копирования, вставки и вырезания объектов. Вызов этих операций должен происходить в контекстном меню или с помощью комбинаций клавиш `Ctrl-C`, `Ctrl-V` и `Ctrl-X`.