Домашнее задание #1: Подключение библиотек. Библиотека raylib.

Все задачи нужно решить обязательно с использование только одной сторонней библиотеки – raylib.

Задача 1. Создание библиотек

В директории `miptlib` лежит исходный код простой библиотеки с одноимённым названием, состоящей из трёх заголовочных `.hpp` файлов и трёх файлов исходного кода `.cpp`. Также там лежит код простейшей программы `test.cpp` для тестирования библиотеки.

(a) Создайте статическую библиотеку из этого исходного кода. То есть файлы `algebra.cpp`, `geometry.cpp` и `statistics.cpp` нужно превратить в один файл статической библиотеки `libmiptlib.a`.

(b) Скомпилируйте программу `test.cpp`, подключив к ней статическую библиотеку `libmiptlib.a`.

(c) Создайте динамическую библиотеку из изначального исходного кода. То есть файлы `algebra.cpp`, `geometry.cpp` и `statistics.cpp` нужно превратить в один файл динамической библиотеки `libmiptlib.so`. Используйте флаг `-fPIC` при компиляции объектных файлов для динамической библиотеки. Если вы на Windows, то динамическая библиотека будет состоять из двух файлов: `miptlib.dll` и `miptlib.dll.a`.

(d) Скомпилируйте программу `test.cpp`, подключив к ней динамическую библиотеку. Запустите скомпилированную программу. Библиотека должна подключаться при запуске программы.

(e) Измените файл `test.cpp` так, чтобы библиотека подключалась не при запуске программы, а спустя две секунды после запуска. Используйте `dlfcn.h`. На Windows эту подзадачу желательно делать, используя компилятор MinGW.

Для решения этой задачи создайте файлы `01a.txt`, `01b.txt`, `01c.txt`, `01d.txt` и `01e.txt`. В которых нужно написать вашу ОС, компилятор, оболочку а также все команды, необходимые для выполнения соответствующей подзадачи. Для подзадачи `(e)` нужно также добавить файл `01e_test.cpp`, содержащий новую версию файла для тестирования.

Задача 2. Цикл

Напишите программу, которая будет рисовать круг, движущийся с постоянной скоростью по горизонтали слева вправо. После того, как круг выходит за границы экрана с правой стороны, он должен появиться с левой стороны.

Задача 3. Вращающийся квадрат

Напишите программу, которая рисует вращающийся квадрат. Квадрат должен вращаться вокруг одной из своих вершин, а не вокруг своего центра.

Задача 4. Движение по кругу

Напишите программу, которая будет рисовать круг, двигающийся по окружности.

Задача 5. Координаты мыши

Напишите программу, которая будет рисовать на экране текст, в котором содержатся координаты мыши в системе координат окна. При изменении положения курсора мыши, текст должен меняться. Используйте функцию `std::to_string` для конвертации в строки.

Задача 6. Столкновение курсора с прямоугольником

Напишите программу, которая бы рисовала прямоугольник, цвет которого изменялся бы при наведении на него мышью. Если курсор мыши находится вне прямоугольника, то он должен рисоваться зелёным цветом. Если курсор мыши находится внутри прямоугольника, то красным.

Задача 7. Двигающийся кружок

Напишите программу, которая будет рисовать кружок на экране. Кружок должен перемещаться влево, вправо, вниз и вверх если зажата соответствующая клавиша стрелочек. При нажатии на клавишу пробел кружок должен менять цвет на случайный. Кружок должен менять цвет только в момент нажатия на пробел. Цвет не должен меняться каждый кадр при зажатии клавиши пробел.

Задача 8. Притяжение к мыши

Напишите программу, которая будет рисовать кружок на экране. При зажатии левой кнопки мыши (ЛКМ), кружок должен начать двигаться к курсору мыши. Решите задачу в двух вариантах:

a. Кружок должен двигаться к курсору мыши с постоянной скоростью. Если ЛКМ не зажата, то кружок должен оставаться на месте.

b. Кружок должен двигаться к курсору мыши с постоянным ускорением. Если ЛКМ не зажата, то кружок должен продолжать двигаться с постоянной скоростью.

Задача 9. Создание кругов

Напишите программу, в которой можно создавать круги по нажатию левой кнопки мыши. При нажатии ЛКМ в определённом месте окна должен появляться круг белого цвета, причём центр круга должен совпадать с положением курсора мыши. При нажатии клавиши пробел цвет всех кругов должен изменяться на случайный.

Задача 10. Задача n тел

В двумерном пространстве находятся n шариков с различными массами и различными электрическими зарядами. Напишите программу, которая будет моделировать движение таких шариков. Шарики не должны двигаться очень медленно; их движение должно быть заметно во все времена симуляции. Движение шариков должно быть одинаково независимо от того, на какой машине программа будет запускаться.

- Шарики действуют друг на друга силой Кулона:

$$F = \frac{q_1 \cdot q_2}{R}$$

  Где $q_{1,2}$ – заряды шариков, R – расстояние между шариками. Обратите внимание, что сила взаимодействия обратно пропорциональна первой степени расстояния, а не второй. Это правильная формула для силы Кулона в двух измерениях.

- Считайте, что силы гравитации между шариками пренебрежимо малы по сравнению с электрическими силами. Их можно не учитывать.

- Столкновения шариков друг с другом можно тоже не учитывать. Шарики взаимодействуют только посредством силы Кулона.

- На границах окна поставьте стенки. Шарики должны упруго отскакивать от стенок.

- Расчёт ускорений, скоростей и положений всех шариков проводите с шагом в длину кадра Δt. Ограничьте максимальное значение количества кадров в секунду (fps) с помощью функции `SetTargetFPS`.

- Если два шарика подойдут слишком близко друг к другу, то сила взаимодействия может стать очень большой. Это приведёт к большой погрешности в вычислениях из-за того, что Δt больше характерного времени изменения силы взаимодействия между шариками. В результате этого шарики начнут чрезмерно быстро двигаться. Чтобы исправить этот баг просто сделайте так, чтобы сила взаимодействия шариков была равна нулю, если расстояние между шариками меньше некоторой величины.

- Начальные значения масс, зарядов, положений шариков задайте случайным образом из некоторых диапазонов. Начальные значения скоростей равны нулю.

- Цвет шарика должен зависит от его заряда. Положительно заряженные шарики рисуйте красным цветом, а отрицательно заряженные – синим.

- Добавьте возможность добавлять шарики, используя мышь. При нажатии левой кнопки мыши, в том месте, где находится курсор, должен создаваться шарик с маленькой массой и с отрицательным зарядом. При нажатии правой кнопки мыши должен создаваться шарик с очень большой массой и положительным зарядом.

Задача 11. (*) Столкновения шариков и ломаных

Напишите программу, в которой можно было бы создавать движущиеся шарики и статичные ломаные. Шарики должны сталкиваться с краями окна, ломаными и друг с другом. В программе должна быть возможность включать/выключать гравитацию, которая будет действовать на шарики. Также в программе должна быть возможность изменять потери скорости шариков при столкновениях. В программе нужно реализовать следующие взаимодействия:

- При нажатии на ЛКМ без зажатого `LCtrl` должен создаваться шарик радиуса 5 пикселей и со случайными компонентами скорости в диапазоне $[-500, 500]$ пикселей в секунду. Все шарики должны быть одинакового радиуса и одинаковой массы.
- При нажатии на ЛКМ с зажатым `LCtrl` можно создавать произвольную ломаную. Пока `LCtrl` зажат, нужно последовательно выбирать точки ломаной. Сама ломаная при этом отображается серым цветом. После отпускания `LCtrl` ломаная создаётся и движущиеся шарики теперь могут с ней сталкиваться. В программе можно создать множество различных ломаных.
- При нажатии на клавишу `G` должна включаться/выключаться гравитация.
- При нажатии на клавишу `W` скорость всех шариков должна увеличиваться на 20%. При нажатии на клавишу `S` скорость всех шариков должна уменьшаться на 20%.
- Коэффициент *decay* влияет на уменьшение нормальной компоненты скорости при каждом столкновении шариков с любым препятствием. По умолчанию его значение равно 1 (упругие столкновения). При нажатии на клавиши `UpArrow`/`DownArrow` он должен увеличиваться/уменьшаться на 0.01. Коэффициент не должен превышать значения 1.
- При нажатии на клавишу `D` должен удаляться последний созданный шарик.
- При нажатии на клавишу `F` должна удаляться последняя созданная ломаная.
- При нажатии на клавишу `Delete` должны удаляться все шарики и все ломаные.

Необязательные задачи

Задача 1. Случайные перемещения

Напишите программу, которая бы рисовала круг, который бы перемещался в случайное место на экране каждые 2 секунды. При этом, все остальное должно работать как прежде, то есть функцию `sf::sleep` использовать не получится. Используйте классы `sf::Time` и `sf::Clock`.

Задача 2. Изменение цвета

Напишите программу, которая бы рисовала круг, цвет которого зависил бы от положения мыши на экране. Красная компонента цвета круга должна меняться от 0 до 255 при изменении x-компоненты курсора мыши. Синяя компонента должна меняться от 0 до 255 при изменении y-компоненты курсора мыши.

Задача 3. Броуновское движение

Напишите программу, которая будет моделировать броуновское движение частиц. Представьте каждую частицу в виде шарика, который перемещается на случайное небольшое смещение в случайные моменты времени.

Задача 4. 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. Этот момент был более рассмотрен в семинаре #4.
- В папке `../classroom_tasks_solutions/context_menu/` содержится реализация контекстного меню – класс `ContextMenu`. Используйте этот класс чтобы добавить контекстное меню в программу. Оно 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`.