* xcb\_connect (const char \*display,int \*screen) — подключение к X -серверу указанного дисплея , используя данную информацию авторизации . Если конкретный экран на этом сервере является предпочтительным , то INT , на который указывает экран (если не равно нулю ),будет установлен на этом экране ; в противном случае экран будет установлен в 0 . В нашем случае: **xcb\_connect(NULL, NULL).**
* **glfwInit()** - инициализация GLFW-библиотеки.
* **glfwWindowHint(GLFW\_DECORATED, GL\_FALSE)** — добавление/изменение некоторых параметров при создании окна. В нашем случае - убираем границы окна.(По умолчанию, GLFW\_DECORATED имеет значение GL\_TRUE).
* glfwCreateWindow(int width, int height, const char \* title, GLFWmonitor \* monitor, GLFWwindow \* share) — создает окно необходимой длины, ширины, с необходимым названием, на необходимом окне. При создании окна и его OpenGL с glfwCreateWindow , можно указать другое окно, с которым новое окно разделит объекты (текстуры , вершинные и элементов буферов и т.д.) (параметр share). В нашем случае: g**lfwCreateWindow(150, 150, "Simple example", NULL, NULL).**
* glfwSetWindowPos(GLFWwindow \*window, int xpos, int ypos) — устанавливаем позицию нашего окна. В нашем случае: **glfwSetWindowPos(window, 64, 25)**.
* **glfwTerminate()** - эта функция уничтожает все остальные окна и курсоры , восстанавливает все измененные гамма потоки и освобождает любые другие выделенные ресурсы
* glfwMakeContextCurrent(GLFWwindow \*window) - эта функция делает OpenGL контекст указанного текущего окна на вызывающем потоке . Контекст может быть сделан только на одном потоке , и каждый поток может иметь только один текущий контекст. В нашем случае: **glfwMakeContextCurrent(window)**.
* glfwSwapInterval (int interval). Для ускорения отрисовки используются несколько окон. Данная функция позволяет менять интервал их перемены. В нашем случае: **glfwSwapInterval(1)**.
* glfwWindowShouldClose(GLFWwindow \* window) — возвращает значение закрывающего флага нашего окна. В нашем случае: **glfwWindowShouldClose(window).**
* glfwGetFramebufferSize ( GLFWwindow \* window, int \* width, int \* height) - функция возвращает размер в пикселях, фреймбуфера указанного окна. В нашем случае: **glfwGetFramebufferSize(window, &width, &height)**.
* Ratio — соотношение.
* glViewport(GLint x, GLint y, GLsizei width, GLsizei height) — инициализирует координаты центра области. В нашем случае: **glViewport(0, 0, width, height)**.
* glMatrixMode() - позволяет установить, в какой матрице выполняются операции.
* GlLoadIdentity() - заменяет текущую матрицу единичной.
* glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble nearVal, GLdouble farVal) - описывает преобразование, которое производит параллельную проекцию . В нашем случае: **glOrtho(-ratio, ratio, -1.f, 1.f, 1.f, -1.f)**.
* glRotatef(GLfloat angle, GLfloat x, GLfloat y, GLfloat z) — производит порот на angle количество градусов вокруг вектора с координатами(x, y, z). В нашкм случае угол мы считаем ,учитывая координаты курсора, **glRotatef( angle, 0.f, 0.f, 1.f)**.
* glfwPollEvents() - эта функция обрабатывает только те события, которые уже находятся в очереди событий , а затем сразу же возвращается.
* glfwDestroyWindow ( GLFWwindow \* window ) - эта функция удаляет окно.