GLFW input output and picking methods

Computer Graphics
On back ground

- GLFW use threads to manage I/O operations and buffer drawing operations
- Each thread runs a callback function
- Some of the callback functions get parameters from the system
- Callback function order can be changed during the program but some callbacks have priority.
Main Callbacks

- `glfwSetKeyCallback(window, key_callback);`
- `glfwSetMouseButtonCallback(window, mouse_callback);`
- `glfwSetScrollCallback(window, scroll_callback);`
- `glfwSetCursorPosCallback(window, cursor_position_callback);`
- `glfwSetWindowSizeCallback(window, window_size_callback);`
- and more …
void key_callback(GLFWwindow* window, int key, int scancode, int action, int mods)
{
    if(action == GLFW_PRESS || action == GLFW_REPEAT)
    {
        switch (key)
        {
        case GLFW_KEY_ESCAPE:
            glfwSetWindowShouldClose(window,GLFW_TRUE);
            break;
        case GLFW_KEY_RIGHT:
            M = glm::translate(M,vec3(-0.1f,0,0));
            break;
        }
    }
}
void cursor_position_callback(GLFWwindow* window, double xpos, double ypos)
{
    if (glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_RIGHT) == GLFW_PRESS)
    {
        updatePosition(xpos, ypos);
        mouseProcessing(GLFW_MOUSE_BUTTON_RIGHT);
    }
    else
    {
        if (glfwGetMouseButton(window, GLFW_MOUSE_BUTTON_LEFT) == GLFW_PRESS)
        {
            updatePosition(xpos, ypos);
            mouseProcessing(GLFW_MOUSE_BUTTON_LEFT);
        }
    }
}
void window_size_callback(GLFWwindow* window, int width, int height)
{
    glViewport(0,0,width,height);
    relation = (float)width/(float)height;
    resize(relation,NEAR,FAR);
}
int main(int argc, char** argv)
{
    int error;
    if (!glfwInit())
        error = -1;
    GLFWwindow* m_window = glfwCreateWindow(640, 480, title.c_str(), NULL, NULL);
    if (!m_window)
    {
        glfwTerminate();
        error = -1;
    }
    glfwMakeContextCurrent(window);
    glfwSetKeyCallback(window, key_callback);
    glfwSetCursorPosCallback(window, cursor_position_callback);
    glfwSetWindowSizeCallback(window, window_size_callback);
}
As we saw we use double buffering when just one buffer is represented on the screen.
Object on the screen may have the same color or may change their color.

When we want to pick one object using the mouse we have to much the current pixel an object.

We use a new Shader to render the scene on the back buffer. Each object in a specific color that will never change. We don’t swap the buffers.

Then we can much the current pixel an object according to its color
When you render on the back buffer for color picking:

- Don’t forget to clean the buffers
  ```c
  glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
  ```
- Don’t forget to turn off the lights
- You can use `glGetIntegerv(GL_VIEWPORT,viewport);` to get window size.
- You can read also the depth of the pixel:
  ```c
  glReadPixels(x,viewport[3]-y,1,1,GL_DEPTH_COMPONENT,GL_FLOAT,&zz);
  ```
Ray Picking

- Ray picking is the process of shooting a line (ie. "ray") from the camera through the 2D viewscreen (where the 3D scene is projected on) into the scene until it hits an object.
- This method can run faster than Color Picking but we have to use optimizations (search for intersection of the ray with all triangle in the scene is not efficient)
Ray Picking Optimizations

- Dividing our big mesh to smaller ones
- Calculating approximated Convex Hull
- Bounding boxes or axis aligned bounding boxes
- Octree