# 计算机图形学第一次作业实验报告


- ID：999（旁听）
- 姓名：袁保杰
- 学号：PB21111714


## 实验情况

本实验要求在已有框架的基础上，写一个交互式画图小程序 MiniDraw，要求画直线 (Line)，椭圆 (Ellipse)，矩形 (Rectangle)，多边形 (Polygon) 等图形元素（图元）。

我仅实现了必做部分，即 `Ellipse` 和 `Polygon` 的实现，演示见 `result.gif`。

## 实验过程

### 添加按钮

直接向 `minidraw_window.cpp` 添加以下代码即可（需要补上对应的函数声明）：

```
        ImGui::SameLine();
        if (ImGui::Button("Ellipse"))
        {
            std::cout << "Set shape to Ellipse" << std::endl;
            p_canvas_->set_ellipse();
        }
        ImGui::SameLine();
        if (ImGui::Button("Polygon"))
        {
            std::cout << "Set shape to Polygon" << std::endl;
            p_canvas_->set_polygon();
        }
```

## 实现 `Ellipse`

根据实验文档，在 `src/assignments/1_MiniDraw/shapes` 中添加 `ellipse.cpp` 和 `ellipse.h` 。椭圆的绘制使用 imgui 提供的 API，定义于 https://github.com/ocornut/imgui/blob/master/imgui.h#L3135：

```
    IMGUI_API void  AddEllipse(const ImVec2& center, const ImVec2& radius, ImU32 col, float rot = 0.0f, int num_segments = 0
```

需要注意：

- `radius` 不能为负数，否则在自动计算 `num_segments` 时会出问题，进而导致图形渲染错误
- `num_segments` 是用直线逼近椭圆的 `segment` 数量，取 `0` 即可自动计算

其他部分与 `Rect` 类似，直接抄写即可。

## 实现 `Polygon`

同样根据实验文档，在 `src/assignments/1_MiniDraw/shapes` 中添加 `polygon.cpp` 和 `polygon.h` 。

`Polygon` 的实现与其他形状有较大不同，主要类定义　下：

```
class Polygon : public Shape
{
    public:
        Polygon() = default;

        Polygon(
            float start_point_x,
            float start_point_y)
        {
            // Start Point
            points_.push_back(ImVec2(start_point_x, start_point_y));

            // Current Point
            points_.push_back(ImVec2(start_point_x, start_point_y));
        }

        virtual ~Polygon() = default;

        void draw(const Config& config) const override;

        void update(float x, float y) override;

        void add_control_point(float x, float y);

        void end_drawing();

    private:
        std::vector<ImVec2> points_;
};
```

具体实现可见代码，简略解释　下：

- 私有变量 `points_` 维护了这个多边形的所有顶点
- `draw` 用于将点相连，描绘多边形
- `update` 用于在绘图过程中，随鼠标更新最后的顶点坐标
- `add_control_point` 用于在绘图过程中，按下左键时，向 `points_` 添加顶点
- `end_drawing` 用于在绘图过程中，按下右键时，向 `points_` 添加起始顶点坐标，闭合多边形

接下来考虑鼠标事件的实现：

- 描绘多边形时，当 `draw_status_` 为 `true` 时，点按鼠标左键，不应该再像描绘其他图形一样，将 `draw_status_` 置为 `false` ，并做一些清理。
  - 需要调用 `add_control_point` 来添加顶点。这里使用 `static_pointer_cast` 将指针类型从基类转到派生类，可能存在危险，不知道有没有更好的解决方法。
  - 代码实现大致　下：

```
    if (draw_status_)
    {
        switch (shape_type_)
        {
            case USTC_CG::Canvas::kPolygon:
            {
                // FIXME: 感觉有点丑陋
                static_pointer_cast<USTC_CG::Polygon>(current_shape_)->add_control_point(start_point_.x, start_point_.y)
                break;
            }

            default:
            {
                draw_status_ = false;
                if (current_shape_)
                {
                    shape_list_.push_back(current_shape_);
                    current_shape_.reset();
                }
                break;
            }
        }
    }
```

- 需要添加鼠标右键的处理，调用 `end_drawing` 来闭合多边形，实现大致　下：

```
void Canvas::mouse_right_click_event()
{
    if (draw_status_)
    {
        switch (shape_type_)
        {
            case USTC_CG::Canvas::kPolygon:
            {
                draw_status_ = false;
                if (current_shape_)
                {
                    // FIXME: 感觉有点丑陋
                    static_pointer_cast<USTC_CG::Polygon>(current_shape_)->end_drawing();
                    shape_list_.push_back(current_shape_);
                    current_shape_.reset();
                }
                break;
            }

            default: break;
        }
    }
}
```

# Extra：环境配置

由于使用 NixOS 作为开发环境，使用 nix-shell 来进行环境配置，书写下面的 `shell.nix` 文件：

```
{ pkgs ? import <nixpkgs> {} }:

pkgs.mkShell {
  name = "USTC_CG_2025_HW01_devshell";

  nativeBuildInputs = with pkgs; [
    cmake
    pkg-config
  ];

  buildInputs = with pkgs; [
    libGL
    libxkbcommon
    xorg.libX11
    xorg.libXrandr
    xorg.libXinerama
    xorg.libXcursor
    xorg.libXi
    xorg.libXext
    xorg.libXxf86vm
  ];

  # OpenGL runtime dependency is not written in the compiled executable
  LD_LIBRARY_PATH = pkgs.lib.makeLibraryPath (with pkgs; [
    libGL
    libxkbcommon
    xorg.libX11
    xorg.libXrandr
    xorg.libXinerama
    xorg.libXcursor
```

```
    xorg.libXi
    xorg.libXext
    xorg.libXxf86vm
  ]);
}
```

为了取消 Wayland 支持，在 `CMakeLists.txt` 中添加：

```
set(GLFW_BUILD_WAYLAND False)
```

运行以下命令完成编译：

```
$ nix-shell
$ cmake -B build
$ cmake --build build
```