Object-Oriented Programming:
GUI - raylib

Lectured by Ming-Te Chi 紀明德

First Semester, 2022

Computer Science Department
National Chengchi University

Slides credited from 李蔡彥 and 廖峻鋒

outline

- raylib
  - Game loop
  - Input
  - Draw
  - 3D-camara and model
  - Examples
- raylib-cpp: a C++ wrapper library for raylib
- For web

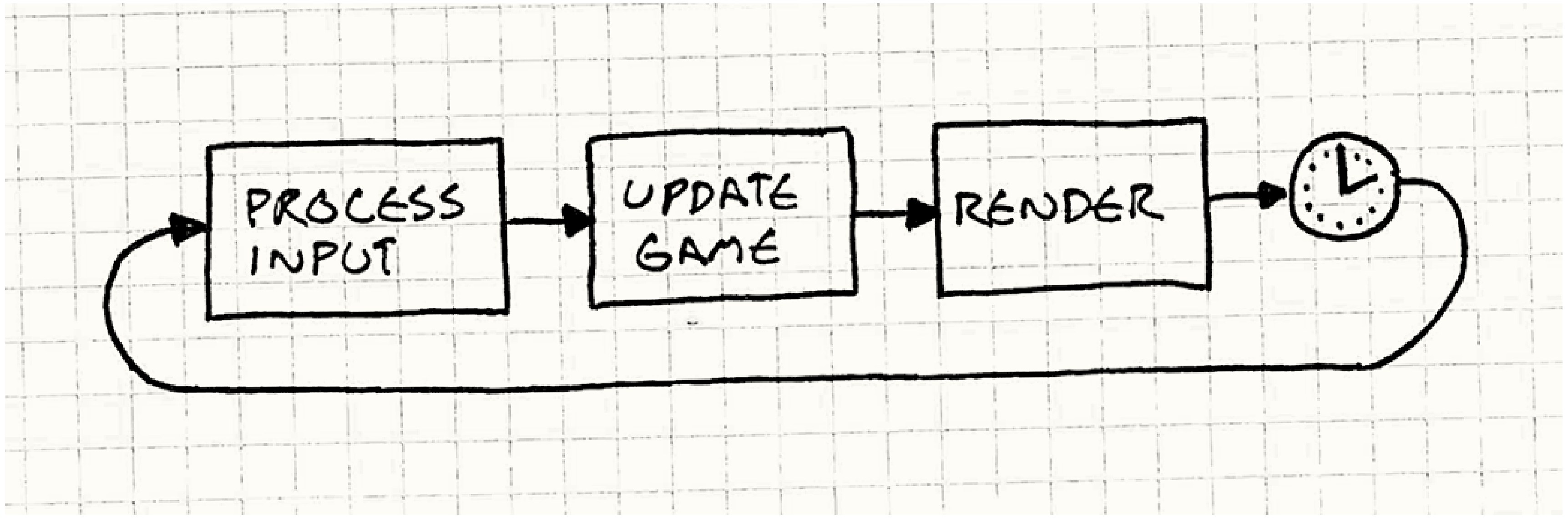
Raylib

- a simple and easy-to-use library to enjoy videogames programming.

Feature

- - NO external dependencies, all required libraries included with raylib
- - Multiplatform: Windows, Linux, MacOS, RPI, Android, HTML5... and more!
- - Written in plain C code (C99) in PascalCase/camelCase notation
- - Hardware accelerated with OpenGL (1.1, 2.1, 3.3, 4.3 or ES 2.0)
- - Unique OpenGL abstraction layer: rlgl

Game loop

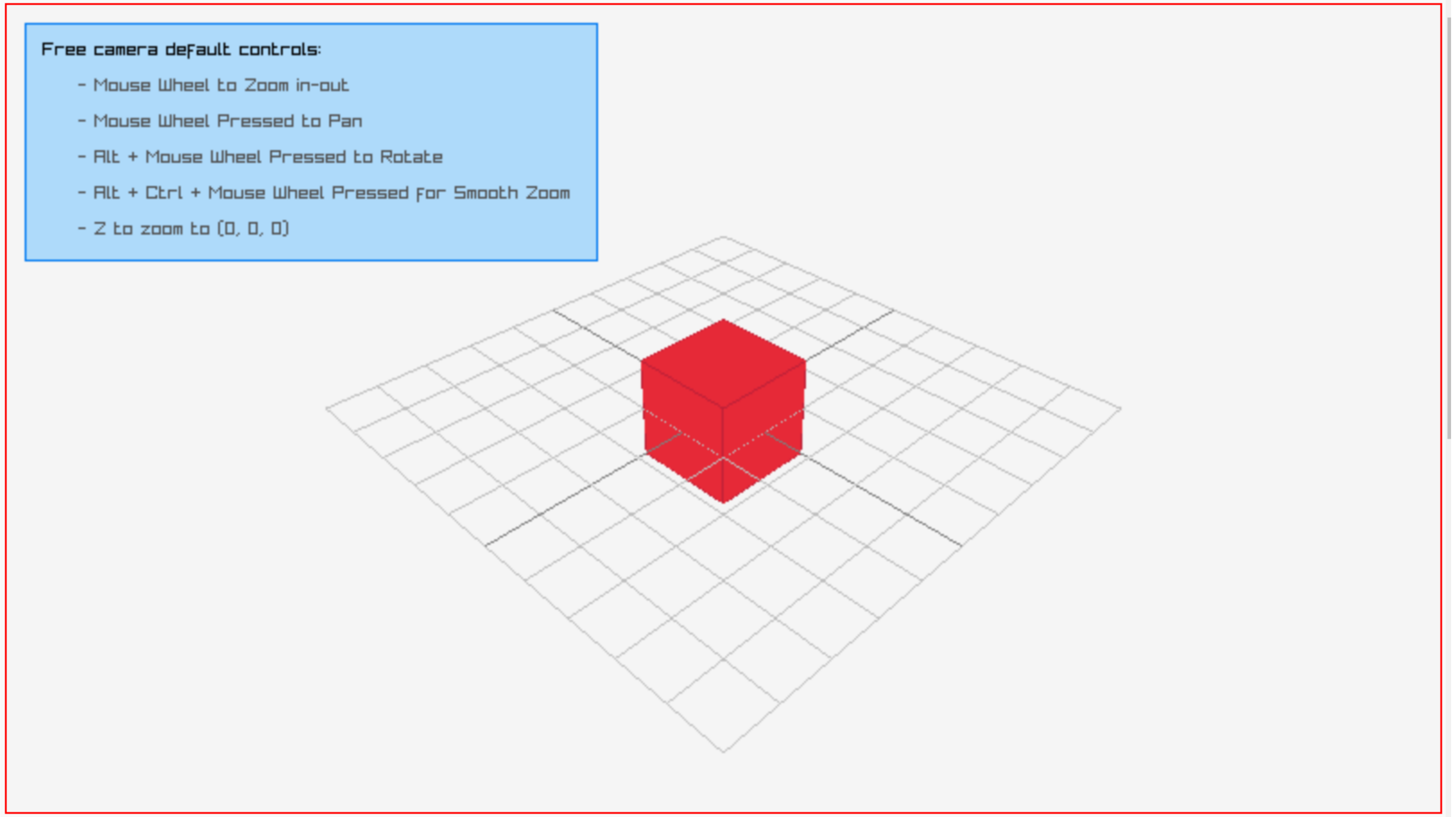


http://gameprogrammingpatterns.com/game-loop.html

```c
#include "raylib.h"
int main(void) {
const int screenWidth = 800;
const int screenHeight = 450;
InitWindow(screenWidth, screenHeight, "raylib [core] example - basic window");
SetTargetFPS(60);

// Main game loop
while (!WindowShouldClose()) // Detect window close button or ESC key
{
// TODO: Update your variables here

    BeginDrawing(); // Draw
    ClearBackground(RAYWHITE);
    DrawText("Congrats! You created your first window!", 190, 200, 20, LIGHTGRAY);
    EndDrawing();
}

CloseWindow();
return 0;
}
```

Input and Draw a circle

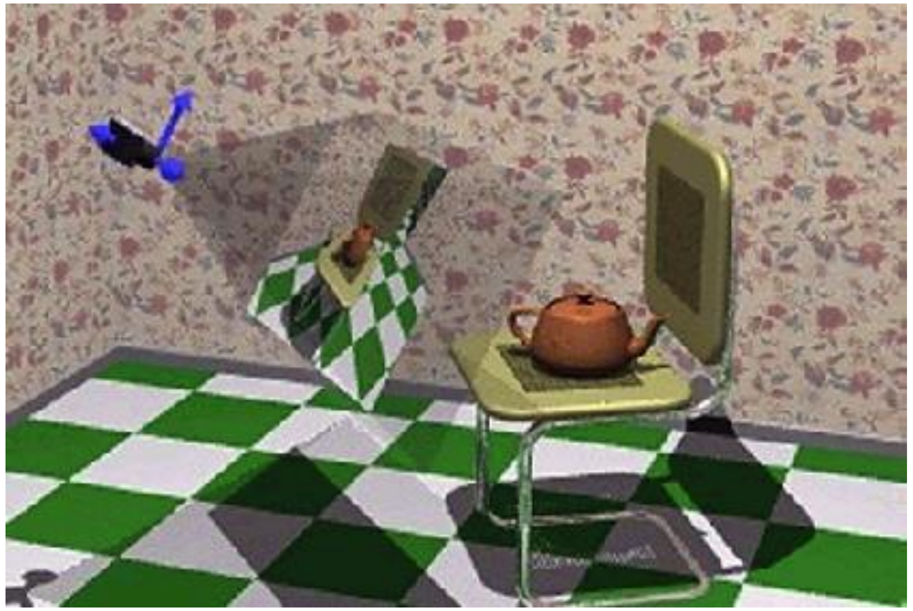
```c
while (!WindowShouldClose()) // Detect window close button or ESC key
{
    // Update
    if (IsKeyDown(KEY_RIGHT)) ballPosition.x += 2.0f;
    if (IsKeyDown(KEY_LEFT)) ballPosition.x -= 2.0f;
    if (IsKeyDown(KEY_UP)) ballPosition.y -= 2.0f;
    if (IsKeyDown(KEY_DOWN)) ballPosition.y += 2.0f;

    // Draw
    BeginDrawing();
    ClearBackground(RAYWHITE);
    DrawText("move the ball with arrow keys", 10, 10, 20, DARKGRAY);
    DrawCircleV(ballPosition, 50, MAROON);
    EndDrawing();
}
```



Free camera default controls:

- Mouse Wheel to Zoom in-out
- Mouse Wheel Pressed to Pan
- Alt + Mouse Wheel Pressed to Rotate
- Alt + Ctrl + Mouse Wheel Pressed for Smooth Zoom
- Z to zoom to (0, 0, 0)

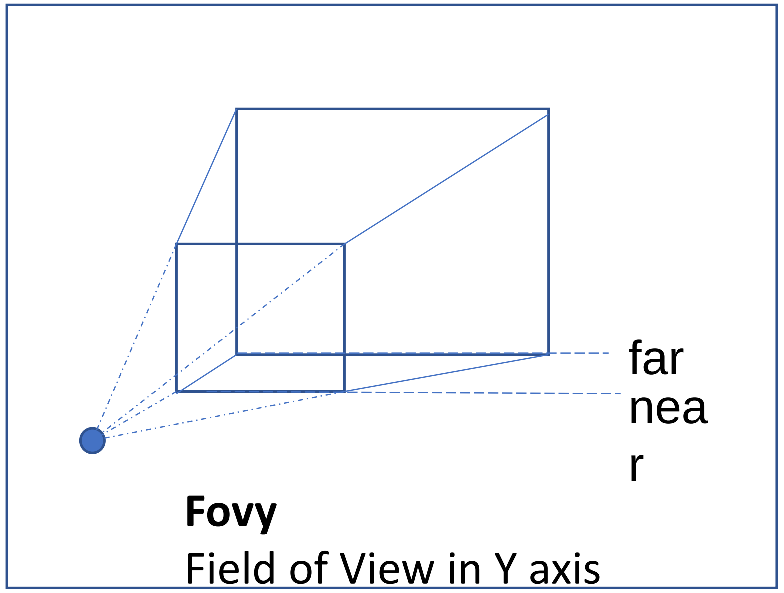
3D (1/2) – camera



```
// Define the camera to look into our 3d world
Camera3D camera = { 0 };
camera.position = (Vector3){ 10.0f, 10.0f, 10.0f }; // Camera position
camera.target = (Vector3){ 0.0f, 0.0f, 0.0f }; // Camera looking at point
camera.up = (Vector3){ 0.0f, 1.0f, 0.0f }; // Camera up vector (rotation towar
target)
camera.fovy = 45.0f; // Camera field-of-view Y
camera.projection = CAMERA_PERSPECTIVE; // Camera mode type
Vector3 cubePosition = { 0.0f, 0.0f, 0.0f };
SetCameraMode(camera, CAMERA_FREE); // Set a free camera mode
```

camera3D

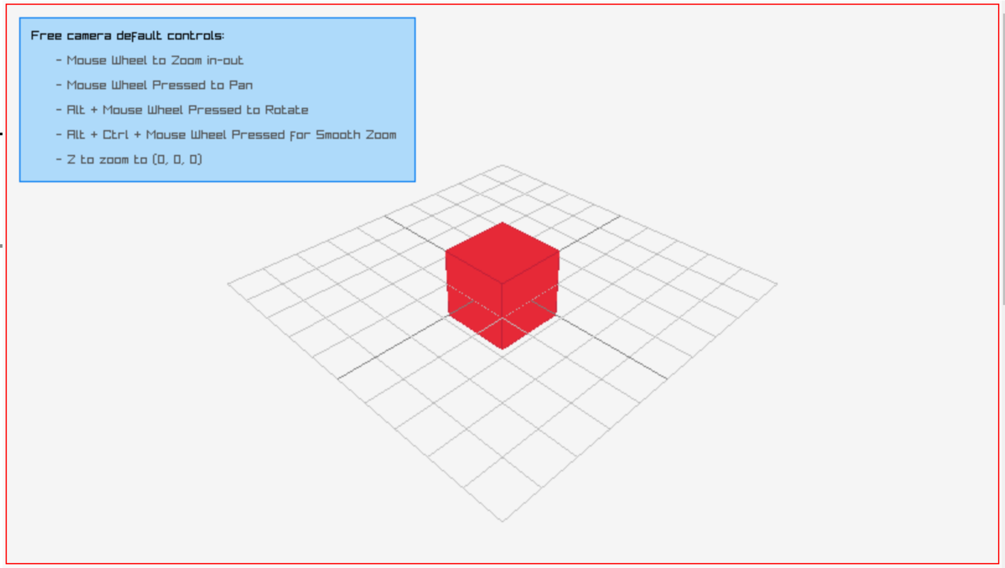
- Source code in raylib.h



```
    // Camera type, defines a camera position/orientation in 3d space
    typedef struct Camera3D {
        Vector3 position;       // Camera position
        Vector3 target;         // Camera target it looks-at
        Vector3 up;             // Camera up vector (rotation over its axis)
        float fovy;             // Camera field-of-view apperture in Y (degree
in perspective, used as near plane width in orthographic
        int type;               // Camera type, defines projection type:
CAMERA_PERSPECTIVE or CAMERA_ORTHOGRAPHIC
    } Camera3D;
```

3D (2/2) – draw 3D scene

```
// Draw
//---------------------------------------------
BeginDrawing();
ClearBackground(RAYWHITE);

BeginMode3D(camera);
   DrawCube(cubePosition, 2.0f, 2.0f, 2.0f, RED);
   DrawCubeWires(cubePosition, 2.0f, 2.0f, 2.0f, MAROON);
   DrawGrid(10, 1.0f);
EndMode3D();

DrawRectangle( 10, 10, 320, 133, Fade(SKYBLUE, 0.5f));
DrawRectangleLines( 10, 10, 320, 133, BLUE);
```



Free camera default controls:
- Mouse Wheel to Zoom in-out
- Mouse Wheel Pressed to Pan
- Alt + Mouse Wheel Pressed to Rotate
- Alt + Ctrl + Mouse Wheel Pressed for Smooth Zoom
- Z to zoom to (0, 0, 0)

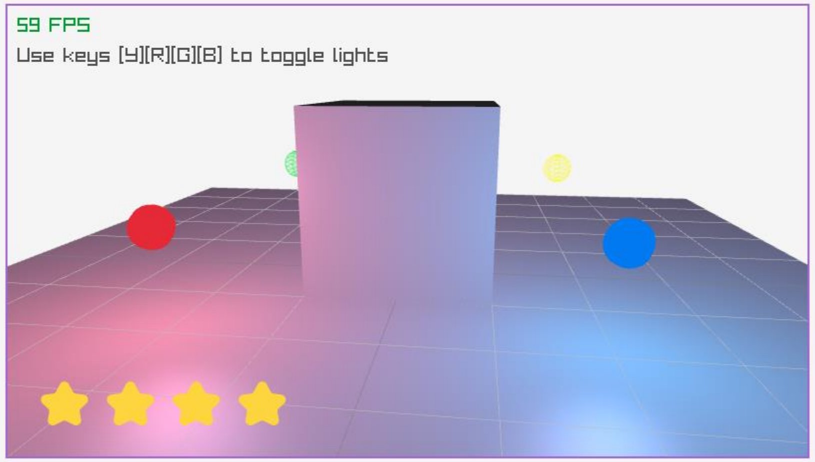
raylib

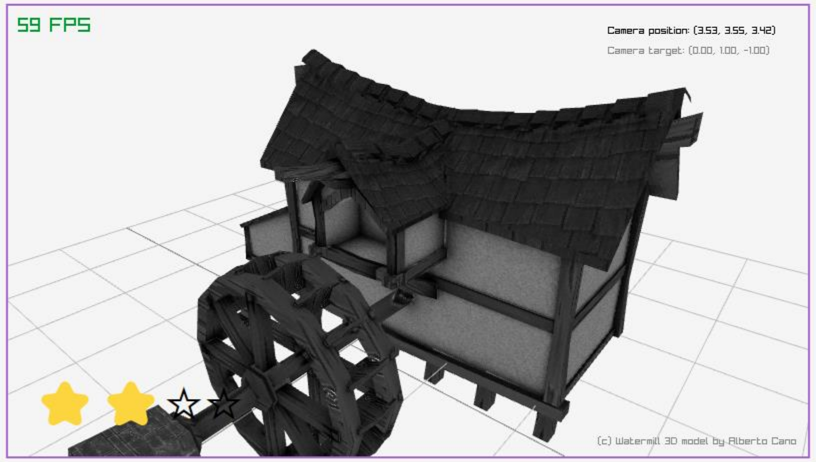
about examples games cheatsheet wiki

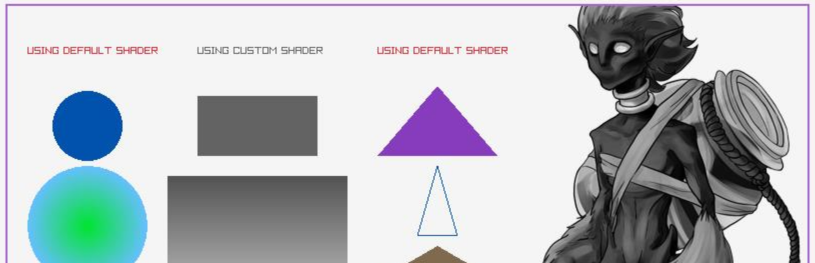
raylib examples are organized by colors depending on the raylib module they focus.

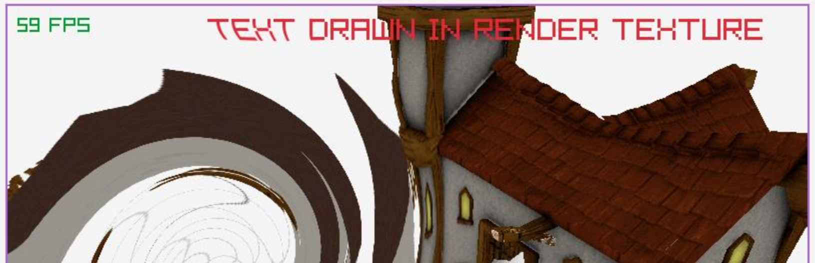
Click on the module to filter examples:

core shapes textures text models shaders audio




59 FPS
Camera position: (3.53, 3.55, 3.42)
Camera target: (0.00, 1.00, -1.00)
(c) Watermill 3D model by Alberto Cano





Raylib-cpp: a C++ wrapper library for raylib

```
   raylib::Window window(screenWidth, screenHeight, "raylib-cpp - basic window");
    raylib::Texture logo("raylib_logo.png");

    while (!window.ShouldClose())
    {
        BeginDrawing();

        window.ClearBackground(RAYWHITE);

        DrawText("Congrats! You created your first window!", 190, 200, 20, LIGHTGRAY);

        // Object methods.
        logo.Draw(
            screenWidth / 2 - logo.GetWidth() / 2,
            screenHeight / 2 - logo.GetHeight() / 2);

        EndDrawing();
    }
```

Raylib-cpp: usage comparison

```
// raylib
Vector2 position(50, 50);
DrawPixelV(position, PURPLE);

// raylib-cpp
raylib::Vector2 position(50, 50);
position.DrawPixel(PURPLE);
```

```
// raylib
DrawTexture(texture, 50, 50, WHITE);

// raylib-cpp
texture.Draw(50, 50, WHITE);
```

```
// raylib
Vector2 position = {50, 50};
Vector2 speed = {10, 10};
position.x += speed.x;
position.y += speed.y;

// raylib-cpp
raylib::Vector2 position(50, 50);
raylib::Vector2 speed(10, 10);
position += speed; // Addition assignment
operator override.
```

Raylib-cpp: source code

```cpp
namespace raylib {
class Vector3 : public ::Vector3 {
 public:
    Vector3(const ::Vector3& vec) : ::Vector3{vec.x, vec.y, vec.z} {}
    Vector3(float x, float y, float z) : ::Vector3{x, y, z} {}

    Vector3(::Color color) {
        set(ColorToHSV(color));
    }
...
 private:
    void set(const ::Vector3& vec) {
        x = vec.x;
        y = vec.y;
        z = vec.z;
    }
};
}  // namespace raylib
```

```c
//In raylib.h

// Vector3, 3 components
typedef struct Vector3 {
    float x;                // Vector x component
    float y;                // Vector y component
    float z;                // Vector z component
} Vector3;
```

Raylib-cpp: DrawCircle3D()

```cpp
namespace raylib {
class Vector3 : public ::Vector3 {
 public:
...
    inline void DrawCircle3D(
            float radius,
            const ::Vector3& rotationAxis,
            float rotationAngle,
            Color color) const {
        ::DrawCircle3D(*this, radius, rotationAxis, rotationAngle, color);
    }
...
};
}  // namespace raylib
```

```c
//In raylib.h
RLAPI void DrawCircle3D(Vector3 center, float radius, Vector3 rotationAxis, float
rotationAngle, Color color); // Draw a circle in 3D world space
```

DrawCircle3D()

```
//In raylib.h
RLAPI void DrawCircle3D(Vector3 center, float radius, Vector3 rotatio
rotationAngle, Color color); // Draw a circle in 3D world space
```

```
//rmodels.c in raylib
// Draw a circle in 3D world space
void DrawCircle3D(Vector3 center, float radius, Vector3 rotationAxis, float rotationAngle,
Color color)
{
    rlPushMatrix();
        rlTranslatef(center.x, center.y, center.z);
        rlRotatef(rotationAngle, rotationAxis.x, rotationAxis.y, rotationAxis.z);

        rlBegin(RL_LINES);
            for (int i = 0; i < 360; i += 10)
            {
                rlColor4ub(color.r, color.g, color.b, color.a);

                rlVertex3f(sinf(DEG2RAD*i)*radius, cosf(DEG2RAD*i)*radius, 0.0f);
                rlVertex3f(sinf(DEG2RAD*(i + 10))*radius, cosf(DEG2RAD*(i + 10))*radius, 0.0f);
            }
        rlEnd();
    rlPopMatrix();
}
```

Working for Web (HTML5)

- emscripten toolchain
  - a complete compiler toolchain to WebAssembly, using LLVM, with a special focus on speed, size, and the Web platform.

```
emcc -c rcore.c -Os -Wall -DPLATFORM_WEB -DGRAPHICS_API_OPENGL_ES2
emcc -c rshapes.c -Os -Wall -DPLATFORM_WEB -DGRAPHICS_API_OPENGL_ES2
...
emar rcs libraylib.a rcore.o rshapes.o rtextures.o rtext.o rmodels.o
utils.o raudio.o
```