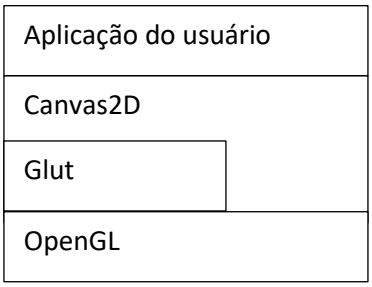
Manual Uso da Canvas2D

Cesar Tadeu Pozzer

2021

A Canvas2D nada mais é que uma API feita em cima das API OpenGL/Glut visando facilitar o desenvolvimento de aplicativos gráficos básicos (figura abaixo). A Glut é uma API destinada à criação da janela da aplicação e faz o processamento dos eventos de mouse/teclado/renderização. A API Opengl não oferece comandos para essas atividades que a Glut desempenha. Além da Glut, outras APIs desempenham o mesmo papel, como por exemplo, a GLFW (com demo disponível no material da disciplina).



O papel da API Canvas2D é simplesmente tornar mais transparente o desenho de primitivas básicas, sem que seja necessário conhecimento do funcionamento das APIs Glut e Opengl. Ela disponibiliza comandos para desenho de linhas, quadrados e pontos. Também permite a seleção de cores e impressão de textos. Por ser baseada no OpenGL, toda programação é baseada em eventos.

Basicamente, na Canvas2D é feito um bypass dos comandos de teclado e mouse, implementados por meio de callbacks (na Glut isso é feito assim, na GLFW é diferente). Esses comandos são agrupados em apenas duas funções, com mais parâmetros. A canvas2D também faz a inicialização do OpenGL, definição da forma de renderização, que nesse caso é em 2D, e especificação de primitivas gráficas simples usando a sintaxe do OpenGL.

A Canvas2D é fundamentada em três funções:

1) mouse() - Trata todos os eventos de mouse. A função é chamada quando ocorrer algum evento de mouse, seja movimento, arrasto, clique ou scrool.
2) keyboard() - Trata todos os eventos de keyboard. A função é chamada quando ocorrer algum evento de teclado.

3) render() – Único local onde devem ser colocados comandos para desenho de primitivas gráficas. Essa função é chamada continuamente. Dependendo da placa de vídeo e das configurações de v-sync, isso pode ocorrer mais de 1000 vezes por segundo.

A forma geral de uso da API é por meio de variáveis globais. Como exemplo, considere a implementação de uma entidade que se move controlada pelo teclado (como no jogo Pac-man). Na função keyboard() devem ser tratados os eventos das setas direcionais que movem a entidade na vertical e horizontal. O seguinte pseudo-código ilustra como deve ser tratado o evento para mover a entidade para a direita.

```
//variável global que controla a movimentação para a direita
bool g_direita = false;

void render()
{
    if (g_direita == true)
        Entidade.x++;   //move a entidade para a direita
    desenhaEntidade();
}

void keyboard(int tecla)
{
    if (tecla == seta_direita)
        g_direita = true;  //muda o estado da variável global
    else
        g_direita = false;  //para de andar para a direita
}
```

Deve-se atentar para não colocar laços de repetição dentro da função render(), como no seguinte exemplo. Isso iria fazer a aplicação travar.

```
void render()
{
    while (tecla == seta_direita)
        Entidade.x++;
}
```

Pode-se usar laços para fazer o desenho de várias entidades, como no seguinte exemplo.

```
void render()
{
    for (linhas)
        for (colunas)
            desenhaEntidade(linha, coluna);
}
```

Para fazer o gerenciamento do que desenhar na canvas2D pode-se adicionar testes condicionais, como no seguinte exemplo, também fazendo uso de variáveis globais.

```
void render()
{
    if (menu == true)
        desenhaMenu();
    if (morreu == true)
        desenhaTelaFimJogo();
    if (jogando == true)
        desenhaCenarioJogo();
}
```

Seguem exemplos de uso:

```
color(1);
translate(100, 100);
point(10, 10); //desenha na posição 110, 110.
rectFill(Vector2(0, 0), Vector2(30, 30));
text(10, 300, “Ola mundo”);
```

A função translate() simplifica o cálculo de coordenadas para desenho de muitos elementos relacionados entre si. Observe que o este comando não é cumulativo, ou seja, chamar translate(10,10), seguido de translate(20,20), não move a origem para coordenada (30,30), mas sim para a coordenada (20,20). Tudo o que vier a ser desenhado após o translate (funções point(), circle(), rect(), etc) sofrerá uma translação de (20,20).

Os seguintes exemplos ilustram o uso do comando translate().

Com	Sem

`translate(200, 200);`
`for(ang = 0; ang < 2*PI; ang += 0.01)`
`{`
`  x = cos(ang) * 50;`
`  y = sin(ang) * 50;`
`  point(x, y);`
`}`
	`for(ang = 0; ang < 2*PI; ang += 0.01)`
	`{`
	`  x = cos(ang) * 50;`
	`  y = sin(ang) * 50;`
	`  point(200 + x, 200 + y);`
	`}`

A inicialização da Canvas2D é bem simples:

```
int screenWidth = 500, screenHeight = 500; //largura e altura inicial da canvas2D

int main(void)
{
    init(&screenWidth, &screenHeight, "Titulo da Janela");
    run();
}
```