DrawPadについて

プログラミング勉強用のグラフィック描画環境です。描画命令によって画面に描画された内容がバッファ上に残っていきます。画面サイズは640×480ピクセル固定で、画面中央が原点(0, 0)、右上方向がプラス方向です。

DrawPadの使い方

Xcodeでプロジェクトファイル DrawPad.xcodeproj を開き、DrawMain.cppファイルのDrawMain()関数の中に、描画などの命令を書きます。

▶ 実行中に利用可能なショートカットキー

・command+Rキー：最初からもう一度実行する。

・command+enterキー：実行を一時停止する。

・command+右矢印キー：描画命令を1つ分ステップ実行する。

DrawPadの命令一覧

グラフィック命令に色を指定する場合は、RGBの各要素を8ビット(0〜255)で表し、16〜23ビット目でRの要素を、8〜15ビット目でGの要素を, 0〜7ビット目でBの要素を表します。色の定数は、kColorBlack, kColorWhite, kColorRed, kColorGreenなど、いくつかの色がDrawing.hppで定義され、Drawing.cppで値が設定されています。必要な色の定数は自分で追加していってください。

1. 基本のグラフィック命令

- `void Clear(int color)`
  指定された色で画面をクリアします。
- `void StartBatch()`
  バッチ処理を開始します。バッチ処理を行っている間、描画命令の実行結果は画面にすぐには反映されなくなります。
- `void EndBatch()`
  バッチ処理を終了します。バッチ処理を行っている間に実行された描画命令の実行結果が、まとめて画面に反映されます。

2. 単純図形の描画

- `void DrawCircle(int cx, int cy, int r, int color)`
  座標(cx, cy)を中心点として半径rの円を描画します。
- `void DrawCircle(int cx, int cy, int r, int color, float a1, float a2)`
  座標(cx, cy)を中心点として、a1ラジアンの角度からa2ラジアンの角度の範囲の円弧を半径rで描画します。
- `void DrawLine(int x1, int y1, int x2, int y2, int color)`
  座標(x1, y1)と座標(x2, y2)を結ぶ線分を描画します。
- `void DrawPoint(int x, int y, int color)`
  指定された座標に点を描画します。
- `void DrawRect(int x, int y, int width, int height, int color)`
  座標(x, y)からサイズ(width, height)の矩形の枠線を描画します。
- `void DrawTriangle(int x1, int y1, int x2, int y2, int x3, int y3, int color)`
  3点の座標を結ぶ三角形を描画します。
- `void FillCircle(int cx, int cy, int r, int color)`
  座標(cx, cy)を中心点として半径rの円を塗りつぶします。
- `void FillCircle(int cx, int cy, int r, int color, float a1, float a2)`
  座標(cx, cy)を中心点として、a1ラジアンの角度からa2ラジアンの角度の範囲の扇形を半径rで描画します。

- `void FillRect(int x, int y, int width, int height, int color)`
  座標(x, y)からサイズ(width, height)の矩形を塗りつぶします。
- `void FillTriangle(int x1, int y1, int x2, int y2, int x3, int y3, int color)`
  3点の座標を結ぶ三角形を塗りつぶします。
- `int GetColor(int x, int y)`
  指定された座標のピクセル値(RGB)を取得します。xは0以上639以下、yは0以上。
- `void Paint(int x, int y, int paintColor, int borderColor)`
  指定された座標からスキャンを開始して、borderColorの色で囲まれた領域を、paintColorの色で塗りつぶします。
- `void Scroll(int x, int y)`
  X方向にxピクセル、Y方向にyピクセルだけ画面データをずらします。ずれた箇所は黒になります。xとyにはマイナスの値もプラスの値も指定できます。

3. テキスト描画の命令

- `void DrawCharacter(char c, int x, int y, int color)`
  座標(x, y)が左下に来るように、文字cを指定された色で描画します。文字のサイズは12x20ピクセル固定です。
- `void DrawText(const char *str, int x, int y, int color)`
  座標(x, y)が左下に来るように、文字列strを指定された色で描画します。文字のサイズは12x20ピクセル固定です。文字列に改行文字が含まれる場合、2行目以降は下方向に描画され、そのY座標はy-20, y-40, ...となっていきます。

4. ユーザのデバイス入出力の取得

- `bool CheckKey(unsigned int key)`
  指定されたキーマスクに該当するキーが押されているかどうかをリターンします。キーを指定するための定数として、ASDWの各キーに対応した定数kKeyA, kKeyS, kKeyD, kKeyW、矢印キーに対応した定数kKeyUpArrow, kKeyDownArrow, kKeyLeftArrow, kKeyRightArrowが用意されています。定数を増やして、MyViewController.mmのkeyDown:メソッドとkeyUp:メソッドを変更することで、他のキーも使えるようになります（shiftキーなどの特殊なキーを除く）。
- `bool CheckMouse()`
  マウスの左ボタンが押されているかどうかをリターンします。
- `int GetMouseX()`
  マウスのX方向のカーソル位置をリターンします。
- `int GetMouseY()`
  マウスのY方向のカーソル位置をリターンします。

5. それ以外の命令

- `void DrawPattern(unsigned int *buffer, int x, int y)`
  指定された座標にグラフィック・パターンを描画します。グラフィックパターンはunsigned int型の配列で、先頭2つの要素がX方向の大きさとY方向の大きさ、その後はARGB,ARGB,ARGB,...となります。ARGBの各要素は、0x00～0xffの8ビットで表されます。A（アルファ値）のデータは、0だと不透明、それ以外だと透明なピクセルとして扱われます。
- `void Sleep(float sec)`
  指定された秒数だけスリープして、処理を中断します。