リファクタリング演習

ゲームプログラミングA #04

向井 智彦

おさらい: 参照とポインタ

- 参照: 変数に別名を与える
  - 基本型でもクラスでも構造体でも使用可
  - 値渡し ＝ データのコピー/クローンを関数に渡して元データを操作させない
  - 参照渡し ＝ 別名を関数に渡して元データに直接読み書きさせる
- ポインタ: 変数に別名を与える + その他
  - 実体はオブジェクトのメモリ位置
  - 具体的な活用法はゲームプログラミングBにて

今日の流れ

- 解説：blocksのプログラムの流れ
  - GLUTとゲームループ
- GLUT演習
  - ゲーム画面サイズの変更
  - 背景色の変更
  - ブロックをカラフルにする
- リファクタリング演習
  - Vector2の利用
  - カプセル化したクラスの導入
- 応用演習
  - アドバンス課題

今週の演習: blocksのリファクタリング

- ブロック崩しゲームの見た目の動作は変えずプログラムコードをわかりやすく整理する
  - Vector2クラスを利用してベクトルデータを扱う
  - 各構造体をカプセル化したクラスに修正

カプセル化のヒント

- 方針：クラス外から値を直接書き換えるものにのみSetXXXあるいは適切なメンバ関数を用意
  - クラス外からは値を読み取るだけの場合はSetは用意しない
- 方針に関するヒント
  - ブロックの位置は変わらない
  - プレイヤーの位置はマウスで変える
  - ボールの位置はボール自身が変える
  - ボールの進行方向は衝突で変わる
  - 各パーツの大きさは変わらない
  - ブロックのvisibilityはボールとの接触時に一度だけ修正され、その後は自動的に減少する

    →ブロックを消すトリガーとなるメンバ関数 disappear を追加

アドバンス課題

1. 壁クラスWallを作成し、ボールが跳ね返る壁の大きさを自由に変更できるように修正
2. ゲームクリア判定を変更
   - 現在はupdateの度にvisibilityをチェックするという無駄な計算になっている。これを、残っているブロックの数or消したブロックの数をカウントし、それぞれ一定数を超えたらゲームクリアとするようなプログラムに修正
3. ボールと壁の接触判定、ボールとブロックの接触判定、ボールとプレイヤーの接触判定を、全てBallクラスのメンバ関数に変更したうえで、Ballクラスのupdate関数内で呼べ出すように変更