4P-01

ゲームとインタラクティブメディアのプログラミングを容易にするC++ライブラリの開発

鈴木　遼†　　長　幾朗†

†早稲田大学

1　はじめに

昨今のメディアアート制作で用いられるプログラミングツールとして、Java などをベースにビジュアル表現に特化したシンプルな API (Application Programming Interface)を提供する Processing[1]や、C++ API で様々なマルチメディア機能を提供する openFrameworks[2]が挙げられる。本研究ではこれら既存のツールよりも複雑な表現やインタラクションを、より短いコードで記述できる C++ライブラリを開発した。本稿では、本ライブラリの、図形を表現するクラス、C++11 以降の規格を活用する API、省力・効率的な 3D レンダリング、多種の I/O デバイスへの標準対応の 4 つの特徴を説明する。

2 プログラミングを容易にする設計

2.1　図形を表現するクラス

本ライブラリでは、点、線分、円、楕円、三角形、矩形、凸四角形、角丸矩形、任意個の穴を持つ多角形など、特別な二次元図形を表現する 9 種類のクラスを定義した。これらの組み合わせによって、ほとんどの 2D 形状モデルを効率的に表現できる。各クラスは変形、幾何計算、描画、他の図形との交差・包含判定のためのメンバ関数を持つ。例えば、任意の別の図形を引数にとる intersects は両者が平面上で交差しているかを、contains は一方が他方を包含しているかを計算する。図形を画面上に表示したい場合は draw や drawFrame を用い、画像上に図形を書き込む際には write や writeFrame を用いる。setPos や moveBy は指定したベクトルに従って図形を平行移動し、stretched は図形を拡大縮小させる。こういった豊富な機能により、ゲームや GUI におけるビジュアライゼーションやインタラクションのロジックを、図 2 のように本質的な部分にのみ集中してコーディングできる。この例(29LOC)では、Processing や openFrameworks による同等の実装(63LOC, 52LOC)と比較して、コードの行数を半分程度に減らすことができた。

Development of the C++ Library for Making Game and Interactive Media Programming Easier

Ryo SUZUKI†, Ikuro CHO†

†Waseda University

2.2　C++11 以降の規格を活用した API

文字列情報を視覚化するすべての関数において、C++11[3]で導入された Variadic templates を活用することで、従来の C++では煩雑だった型安全な数値→文字列変換を簡易に記述できるようにした。

```
Println(L"Hello", 1+1, Pi);
```

また、失敗する可能性のある関数の戻り値を optional 型で統一して表現することで、一貫性と安全性を両立する API を設計した。

2.3　省力・効率的な 3D レンダリング

Processing など既存のツールを用いた通常の 3D シーンのレンダリングでは、光源の上限数(8 個)が画面上の演出を制限する。本ライブラリでは最大 128 個の点光源が登場するシーンをレンダリングできるよう Deferred Rendering[4]を標準で採用し、必要に応じて G-Buffer の解像度をスケールすることで、タブレット PC のように演算性能が低い実行環境においても 60fps を維持できるようにした。また、Height map をもとに地形のポリゴンメッシュをリアルタイムで生成し、Wang の手法[5]で 2D 画像からボリューメトリックな雲を生成・描画するなどの API を提供することで、図 1 のように、視点が変わっても破綻しない屋外の 3D シーンを、わずかな事前準備とプログラムだけで構築できるようにした。

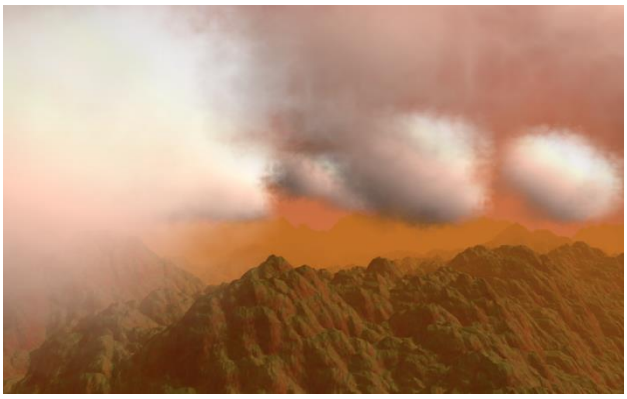

図 1 プログラムで描かれた 3D シーンの一例

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.

```
void Main() {
const Point blockSize(40, 20);
const double speed = 8.0;
Rect wall(60, 10);
Circle ball(320, 400, 8);
Vec2 ballSpeed(0, -speed);
Array<Rect> blocks;
for (int y = 0; y < 5; ++y)
  for (int x = 0; x < 16; ++x)
    blocks.emplace_back(x*blockSize.x, 60 + y*blockSize.y, blockSize);
while (System::Update()) {
  ball.moveBy(ballSpeed);
  wall.setCenter(Mouse::Pos().x, 420);
  for (auto it = blocks.begin(); it != blocks.end(); ++it)
    if (it->intersects(ball)) {
      ((it->bottom.intersects(ball) || it->top.intersects(ball))
      ? ballSpeed.y : ballSpeed.x) *= -1;
      blocks.erase(it);
      break;
    }
  for (const auto& block : blocks)
    block.stretched(-1).draw(HSV(block.y - 40));
  if (ball.center.y<0 && ballSpeed.y<0)
    ballSpeed.y *= -1;
  if ((ball.center.x<0 && ballSpeed.x<0) || (Window::Width()<ball.center.x && ballSpeed.x>0))
    ballSpeed.x *= -1;
  if (ballSpeed.y>0 && wall.intersects(ball))
    ballSpeed = Vec2((ball.center.x - wall.center.x) / 8, -ballSpeed.y).normalized()*speed;
  ball.draw();
  wall.draw();
  }
}
```

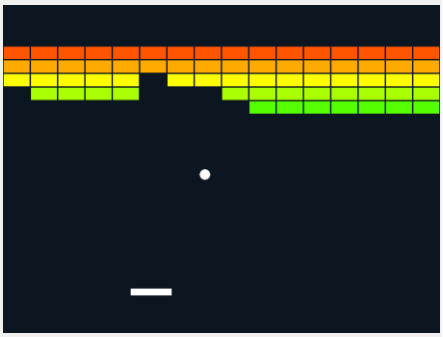

図 2 左:本ライブラリを用いて実装したブロックくずしゲームのソースコード[7] 右:実行結果

2.4 多種の I/O デバイスへの標準対応

ゲームやメディアアートで人気のある入力デバイスを調査し、本稿執筆時点で 12 種類のデバイスに標準で対応した（表 1）。ユーザはこれらのデバイスを扱う際に追加のプラグインや SDK を導入する必要が無くなり、さらに録音音声の FFT や深度カメラのデータのビジュアライズなど、本ライブラリの各種機能と密接に連係する、一貫性のある API を利用できるようにした。

表 1 標準対応した I/O デバイス

マウス、キーボード、Kinect, Kinect v2, Leap Motion, Web カメラ, マイク, Arduino, ペンタブレット, ゲームコントローラ, マルチタッチディスプレイ, Myo

3 評価と今後の課題

本研究で実装されたプログラミングライブラリ[6]は 2,500 回以上ダウンロードされ、Web 上で公開されているユーザ作品は 50 を超えている。将来は本ライブラリをプログラミング教育で活用し、その効果を評価する予定である。また、ユーザビリティのために Processing のようなスタンドアローン IDE の開発を進めている。

4 おわりに

本稿では、図形を表現するクラス、C++11 以降の規格を活用した API 設計、省力・効率的な 3D レンダリング、多種の I/O デバイスへの標準対応によって、ゲームやインタラクティブメディアの作品を、これまでのツールよりも短いコードで容易に実装できる C++ライブラリを説明した。本ライブラリの具体的な効果として、ゲームのビジュアライゼーションとロジックを従来よりも大幅に少ない行数で記述できる例を示した。

参考文献

[1] Processing, https://www.processing.org/

[2] openFrameworks, http://openframeworks.cc/

[3] Du Toit, Stephan. "Working Draft, Standard for Programming Language C++ Technical Report N3337= 12-0027, ISO/IEC JTC 1." *Information Technology Subcommittee SC* 22 (2012).

[4] Koone, R. "Deferred Shading in Tabula Rasa, GPU Gems 3, Nguyen, H." (2007).

[5] Wang, Niniane. "Realistic and fast cloud rendering." *Journal of graphics tools* 9.3 (2004): 21-40.

[6] Play Siv3D!, http://play-siv3d.hateblo.jp/

[7] Play Siv3D! Examples, http://play-siv3d.hateblo.jp /entry/jp/example/breakout (2015/1/7 アクセス)

Copyright ©2015 Information Processing Society of Japan.
All Rights Reserved.