SceneGraphを用いたゲームプログラムの為のテスト作成手法

055722G 小林 佑亮　指導教員：河野真治

1　研究背景・目的

当研究室では学生実験において、PlayStation3 を用いた並列的なゲームプログラムの作成を行っている。そこで使用されるのが当研究室で開発した Cerium レンダリングエンジンである。Cerium を用いたゲームプログラムの例題として PlayStation2 で動作していた超弾帝 (スーパーダンディ) というゲームの移植を行った。

その際、オブジェクトの描画、衝突判定などで不具合が発生した。超弾帝には約 100 種類に及ぶオブジェクトが存在し、今後も同様な不具合を修正していく必要があると考えられる。

本研究ではゲームプログラムを SceneGraph 単位でテストすることでオブジェクトごとの振る舞いや描画をチェックする。これにより、ゲームのデバッグを容易にし、今後の学生実験におけるゲームの移植や改良、作成を円滑にすることができる。

2　SceneGraph によるゲームフレームワーク

ゲームの中の一つの場面 (Scene) を構成するオブジェクトやその振る舞い、ゲームのルールの集合を SceneGraph とする。SceneGraph のノードは親子関係を持つ Tree で構成される。(図 1)

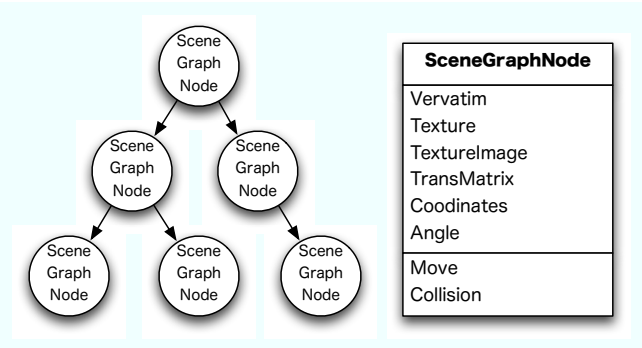

図 1: SceneGraph

親子関係とは、親オブジェクトの回転や並行移動等の行列計算による頂点座標の変更が子オブジェクトにも反映する関係の事である。これは子に対してスタックに積まれた親の変更行列を掛ける事で実現できる。

ゲーム内で使用するオブジェクトは Blender というオープンソースの 3 次元コンピュータグラフィックスソフトウェアで作成する。作成したオブジェクトモデルは python スクリプトにより xml ファイルに変換され、それを元に SceneGraph が生成される。

生成した SceneGraph の各ノードには、オブジェクトの動きを決める `move` という関数と、オブジェクトの相互作用を決める `collision` という関数を入れる変数がある。この変数は、`set_move_collision` という関数によって変更される。

2.1　move

以下の例では、`boss1_move_right` は、ボス 1 を右に `node->stack_xyz[0]` だけ移動させる。スクリーン適当な位置にくると、`move` をボス 1 を左に移動される関数 `boss1_move_left` に設定する。

```
static void
boss1_move_right(SceneGraphPtr node, int screen_w, int screen_h) {
  node->xyz[0] += node->stack_xyz[0];
  if(node->xyz[0] > screen_w-280) {
    node->set_move_collision(boss1_move_left, boss1_collision);
  }
}

static void
boss1_move_left(SceneGraphPtr node, int screen_w, int screen_h) {
  node->xyz[0] -= node->stack_xyz[0];
  if(node->xyz[0] < 280) {
     node->set_move_collision(boss1_move_right, boss1_collision);
  }
}
```

この `move` では、ボス 1 が常に画面内に描画されている必要がある。これはテストすべき条件の一つとなる。

2.2　collision

この変数は、オブジェクトの衝突判定する関数を入れる。例えばシューティングゲームなら自機と敵、または、弾を引数に持ち、当たった時のダメージや、ゲームオーバーの判定を行なう。判定にしたがって、`set_move_collision` により新しい `move,collsion` を設定する。これは、State Pattern と呼ばれるパターンである。

3 CppUnit を用いた SceneGraph のテスト

CppUnit は、C++の単体テストを自動化する frame work である。 CppUnit の特徴は、テストケースを増やす事が容易であり、1 つの事象に対して様々なテストケースを同時にテストする事が出来る。また、羅列したテストケースは一括で実行、結果の表示ができる。(図 2)

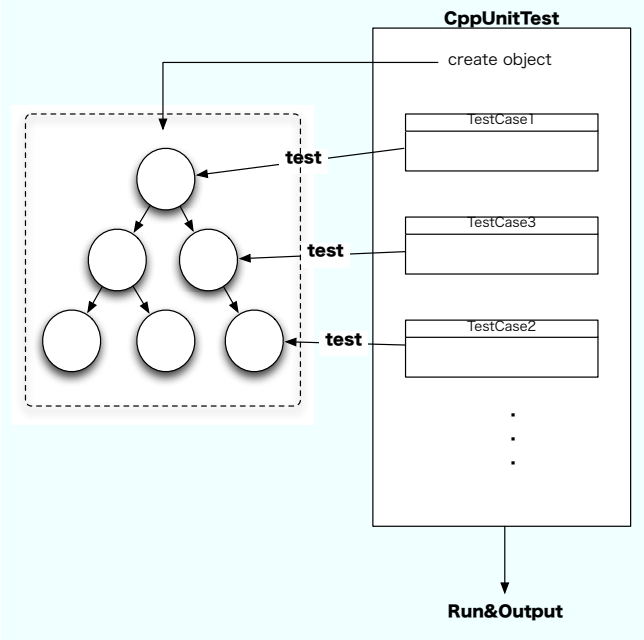

図 2: CppUnitTest

3.1 ゲームプログラムのテスト

3 つの SceneGraph を持つオブジェクトのテストを行った。このオブジェクトは本体の他に左右にパーツを 1 つずつ持つ。本体を Tree の root として左右のパーツがその子供になっている。(図 3)

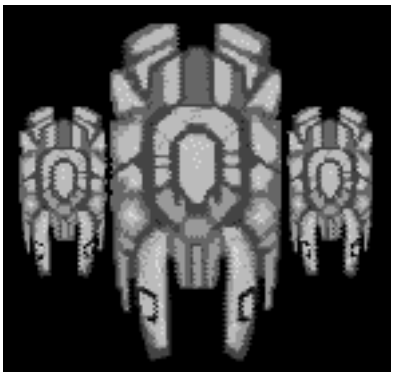

図 3: Boss1

ここで、各オブジェクトの SceneGraph はその親や子、兄弟に対するアドレスを保持している。従って、パーツオブジェクトのテストを行いたい場合は、そのオブジェクトの Root である本体から辿れば、パーツオブジェクトの各パラメータを参照する事が出来る。

そこで、Root のアドレスを取得する getSGP() という関数を作成した。getSGP() 関数によって取得してきた Root のアドレスを使って、各オブジェクトの座標を取得し、その初期化が正しいか、状態遷移において正しい値を保持しているかテストした。

```
void
sgTest::rootTest() {
  test_init();

  sg_root->print_member();
  CPPUNIT_ASSERT_EQUAL((float)width/2, sg_root->xyz[0]);
  CPPUNIT_ASSERT_EQUAL(0.0f, sg_root->xyz[1]);
  CPPUNIT_ASSERT_EQUAL(-100.0f, sg_root->xyz[2]);
}

void
sgTest::childTest() {
  while (sg_root) {
    if(sg_root->children != NULL) {
      sg_root->children->print_member();
...
      sg_root = sg_root->children;
    } else if(sg_root->brother != NULL) {
      sg_root->brother->print_member();
      CPPUNIT_ASSERT_EQUAL(0.0f, sg_root->brother->xyz[0]);
...
      sg_root = sg_root->brother;
    } else {
...
```

このテストの結果、全てのオブジェクトの位置の初期値が正しいことは確認できた。

Cerium においてもこのテストケースは有効であった。

4 今後の課題

しかし、`move,collision` 中の各オブジェクトの座標は確認出来なかった。よって `move,collision` をテストする為に各 `move,collision` を別々に抜き出してテストする手法を今後実装していく。

参考文献

[1] 宮國 渡, 河野 真治, 神里 晃, 杉山 千秋: Cell 用の Fine-grain Task Manager の実装, 第 108 回システムソフトウェアとオペレーティング・システム研究会 (2008)

[2] 高橋 寿一:知識ゼロから学ぶソフトウェアテスト, 翔泳社 (2005)

[3] 伊藤 喜一:CppUnit 入門, (http://www.ogis-ri.co.jp/otc/hiroba/technical/CppUnit/)