2ZA-8

2次元平面における物体の衝突判定のための空間分割アルゴリズム

鈴木達也† 築地立家‡
東京電機大学 理工学部

1. はじめに

一般的に，2次元空間における broad-phase 衝突判定処理とは，衝突発生の箇所を前処理で得られている木構造データから高速に特定する処理であり，その際に，quad-tree や kd-tree が用いられる[2,3,4,5]．しかしながら，スクロール型の 2D ゲームの多くでみられるように，プレイヤが操作するキャラクタが複雑な形状の壁面をもつ回廊を移動するゲームにおいては，キャラクタと壁面の衝突判定のための，より高速な衝突判定アルゴリズムが考えられる．本研究では，このような場合において，quad-tree や kd-tree を利用した衝突判定アルゴリズムよりも高速に動作するアルゴリズムを提案する．

2. 研究目的

一般に，quad-tree や kd-tree を利用した衝突判定アルゴリズムにおいては，複雑な形状の壁面との衝突判定にかかる時間は，複雑度が増すほどに tree の探索深度が深くなるために，時間がかかってしまう．そこで，本研究では，衝突判定の精度を保ちつつ,判定箇所における壁面の複雑度が変化しても，処理判定の速度が落ちないような空間分割の方法とそれに付随するアルゴリズムを提案する．また，この空間分割は，壁面の線分データをいれただけで自動に行われるようにする．

3. 研究内容

本研究では，図1のように，キャラクタが多数の線分で形成される回廊を移動するときのキャラクタと線分との衝突判定について，既存アルゴリズムおよび本研究で提案するアルゴリズムの効率性についての検証実験を行う．

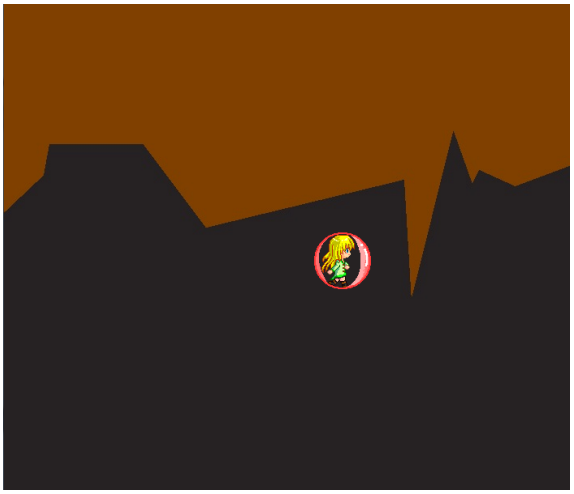

図 1 線分で形成される回廊を移動するキャラクタ

Space Devision Algorithms for 2D-collision detection
†Tatsuya Suzuki ‡Tatsuie Tsukiji
Tokyo Denki University School of Science and Engineering

3.1 衝突判定の narrow-phase

衝突判定の narrow-phase においては，2端点をつないだ線分と球状のキャラクタの中心との距離が球の半径以下になったら（図2において球の中心が水色の領域に侵入）衝突判定を行うようにしている．その際，球の衝突後の速度は，法線ベクトルに対しする入射角と反射角が一致するように算出される．さらに，線分との距離は離れているけれど端点との距離が球の半径以下になった場合（図2におけるピンクの位置）は，その端点を始点とする線分との衝突判定を行う．

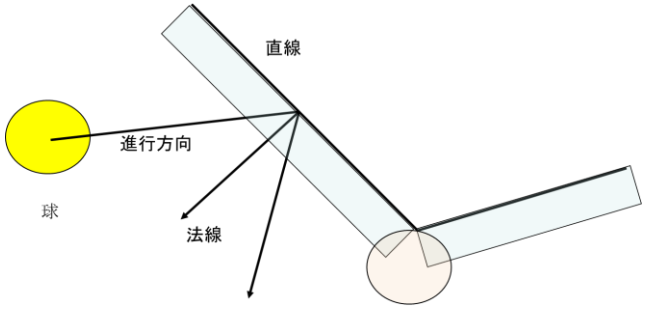

図 2 球と線分の衝突

3.2 空間分割

本研究においては，ゲーム空間となる回廊は単連結であるとする．その境界を閉じたループとして形成する各線分の2端点の2次元座標が入力されたときに，以下のソート処理を行う．任意の線分から開始して，入力された線分をノードとする線形リストを作成する．各ノードの子ノードは，そのノードの終点を端点にもつような線分であり，その端点が子ノードの始点となる．

次に，ソート済みの線分を連続した k 本毎にグループ化する．ただし，各グループの一番最後の線分を一番最初の線分としてグループ化させることで空間の裂け目を無くした．

以下では，線分 L の始点を L^s，終点を L^e と表記して，点 P の x 座標を P_x と表記する．本提案手法では，連続した k 本の線分 A,B,…,F が形成する第 i グループが定義する X 射影空間 X_i と Y 射影空間 Y_i を

$$X_i = \{(x,y) : A^s_{i,x} \leqq x < F^e_{i,x}\},\ Y_i = \{(x,y) : A^s_{i,y} \leqq x < F^e_{i,y}\} \quad (1)$$

と定義する．

3.3 衝突判定の broad-phase

衝突判定の broad-phase においては，キャラクタである球の中心点 Q が現在どの X 射影空間に属しているのかを表1の手続きによって常時モニタリングしておく．

表 1 グループ番号モニタリング

(Q_x,Q_y) ＝ 球の中心の座標
(v_x,v_y) ＝ 球の速度
i＝現在所属している X 射影空間番号
(2)if ($A^s_{i,x}$＜ Q_x && v_x ＜ 0) then i--;
(3)if ($F^e_{i,x}$≧Q_x && v_x ＞ 0) then i++;

そして，空間 X_i に属している場合に衝突が発生したら，第 i グループに属する各線分とのペアリングによって衝突判定を行う．

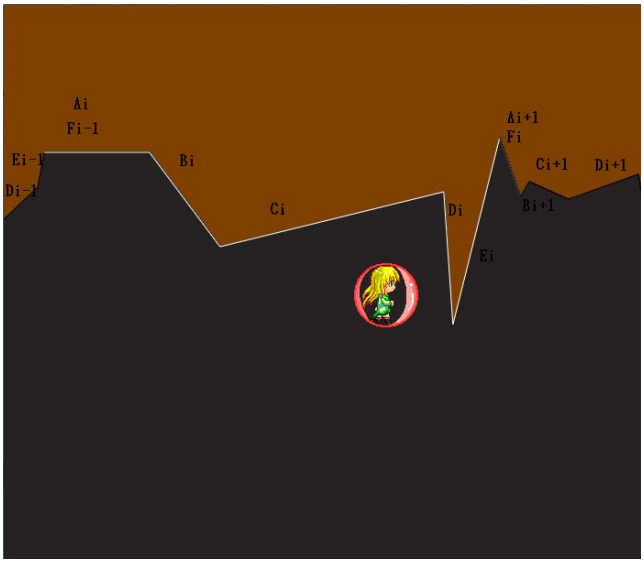

図 3 線分のグループ化

4. 提案アルゴリズムの性能実験

本研究では，表 2 の開発環境のもので，提案アルゴリズムと既存の衝突判定の broad-phase 用のアルゴリズムの性能の実験を行った．ただし，衝突判定の narrow-phase 用のアルゴリズムは，いずれの場合においても，3.1 のものを用いた．

表 2 開発環境

OS	Windows 7 Professional 32bit
プロセッサ	Intel Core2Duo CPU P7550 @ 2.26GHz
実装メモリ	2.00GB
開発環境	Microsoft Visual Studio 2010 Microsoft DirectX

次の表は，衝突判定の broad-phase において，空間分割を一切しない場合, quad-tree を使用して空間均等分割して各空間に 0～k 本の線分が属する場合, kd-dtree を使用して各空間に 0～k 本の線分が存在する場合, および本研究で作成したプログラムの場合，のそれぞれを， 30 秒間の間実行して毎秒毎の FPS 値を観測したグラフである. ただし，FPS 値とは，1 秒間に実行される描画の回数であり，その最大値は 2D ゲームの標準である 62FPS に設定されている．

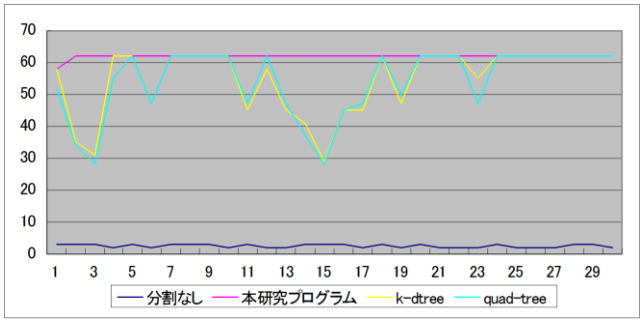

図 43.1 秒ごとの FPS のグラフ

5. 結論・考察

表 3 から，quad-tree や kd-tree を用いて空間を均等分割した場合，たびたび，処理落ちが発生している．一方，本研究のプログラムでは常時最大 FPS を達成しており，処理落ちは発生しなかった．これは，回廊の境界の形状がどんなに複雑であっても，一定の数でグループ化させたことによる．しなわち，座標が密集した部分でも過疎している部分でも一定のかつ少ない処理回数で衝突判定が行われており，結果として，衝突判定による処理落ちを避けることができた．

今後は，より大規模で，かつその境界より複雑な形状の回廊において，本提案アルゴリズムの性能を評価すること，また，キャラクターの数を増やした場合の性能を評価すること，また，キャラクターの移動速度を上げた場合の衝突判定の正確さおよび効率を性能すること，があげられる．

6. 参考文献

[1] Hakuhin HAKUHIN's home page < http://hakuhin.jp/ > (2014/01/12)
[2] 平山 尚 :『ゲームプログラマになる前に覚えておきたい技術』秀和システム pp.675-692 2008
[3] Betlye75 Bently, Jon. “A Modified Algorithm for Convex Decomposition of Associative Searching.” Communications of the ACM. Vol.18, no.9, pp.509-517, 1975.
[4] Friedman77 Friedman, Jerome, Jon Bentley, Raphael Finkel. “An Algorithm for Finding Best Matches in Logarithmic Expected Time” ACMTransactions on Mathematical Software, vol.3, no.3, pp.209-226.1977.
[5] MACDonald90 MacDonald, David, Kellogg Booth. “Heuristics for RayTracing Using Space Subdivision.” The Visual Computer, vol.6, no.3, pp.153-166, 1990