修士論文　　平成 24 年度 (2012)

破片飛散現象表現におけるカメラ回避制御

東京工科大学大学院
バイオ・情報メディア研究科 メディアサイエンス専攻

成田 晃

修士論文　　平成 24 年度 (2012)

破片飛散現象表現におけるカメラ回避制御

指導教員　渡辺 大地

東 京 工 科 大 学 大 学 院
バイオ・情報メディア研究科 メディアサイエンス専攻

成田 晃

論 文 の 要 旨

論文題目	破片飛散現象表現におけるカメラ回避制御
執筆者氏名	成田 晃
指導教員	渡辺 大地
キーワード	破壊、ユーザー制御、物理シミュレーション、リアルタイムシミュレーション

[**要旨**]

近年、コンピュータの性能向上に伴い3DCGを用いたコンテンツも増加している。3DCG分野において物理シミュレーションは、リアルなアニメーションを生成するために重要な手法の一つである。物理シミュレーションによって、剛体や流体などの挙動を様々なパラメータを設定し計算することでリアリティのある動きを容易に表現することができる。

一般に、物理シミュレーション結果にユーザーの意図を反映させるには多くのパラメータ調整が必要であり、出力結果が好ましくない場合が多い。出力結果が望ましくない例として、物体の爆破シミュレーション時の破片飛散現象表現においてカメラの前で重要なアクションが行われているとき、カメラに対して多くの破片が降り注いでしまい、破片しか映らないシーンができてしまうといったことがある。爆発時飛散する破片はシミュレーション開始時に行先を予測することはできず、前記のような事態を回避するようにパラメータを設定するのは困難である。

映画やアニメなどでカメラに破片が飛んでくる場合、画面の上下左右方向に抜けていくことがある。これにより、破片が突然消えることやカメラの周囲に破片が降り注がないといったことにならず、不自然な破片挙動にならず迫力を損なわない。破片が動くカメラに当たらずに、カメラの上下左右方向に逸れるような軌道をとることを目標とする。本稿では、カメラに降り注ぐ破片に対して力を加えることで破片の動きを制御し、カメラから違和感なく逸らすような破片の軌跡を描くリアルタイム剛体シミュレーション手法を提案した。

本研究では、カメラと破片の衝突予測位置から最小限の力を加える事で、ニアクリップ面との干渉しないように破片の挙動を制御を行った。カメラの最大移動速度からカメラと破片が衝突する可能性のある衝突候補となる破片を検出する。衝突候補の放物運動の放物線とカメラの最大移動範囲となる球と交差判定を行い、最短衝突予測時刻を求め、衝突候補が投影面上を通るときの予測位置を検出する。その後、最短衝突予測時刻と衝突予測位置から調整力を計算し衝突候補となる破片に加える。

A b s t r a c t

Title	The fragment avoids camera in fragment dispersion phenomenon expression
Author	Hikaru Narita
Advisor	Taichi Watanabe
Key Words	explosion, user control, physical Simulation, real-time simulations

[**summary**]

In recent years, t he content increased with 3DCG as computer performance improves. Physical simulation is one of important techniques to generate realistic animation in the field of 3DCG. We can easily be expressed with reality in motion by means of a physical simulation to calculate some parameters to configure the behavior of such as rigid and fluid.

In general, output that t o reflect the user's intention is a physical simulation result many parameters need to be adjusted is often undesirable. As an example that undesirable output when there is a important movement in front of the camera in the expression of simulation time explosion fragment scattering phenomenon of the object, there is a lot of fragment that would pour in to the camera, and they can be reflected only scene that does not fragment. Explosion fragments scatter is not possible to predict the destination at the simulation start, setting the parameters to avoid a situation that as above is difficult.

If the fragments flies into the camera, such as movies and anime, you may go through the Vertically and horizontally of the screen. As a result, it will not be not flying fragment around the camera and fragment that suddenly disappears. It does not impair stringency does not become unnatural behavior fragment. The goal is to hit the camera without moving the fragments, such as taking the fragments deviate in the direction of the camera. In this paper, we have proposed a real-time rigid body simulations of fragment trajectory controls the movement of the fragment by applying a force to the fragment falling on the camera, without feeling like distract from the camera.

In this paper, by minimal force and fragment from the position of the camera collision prediction, we have to control the behavior of the fragment so as not to interfere with the near clip plane. We detect candidate fragment collision fragment that could collide with the camera from the maximum speed of the camera. Judgment intersects the sphere which is the maximum range of movement of the camera motion and the parabolic parabolic collision candidate to determine the shortest collision prediction time to detect a collision candidate predicted position when passing through the projection surface. In addition to the fragment collision candidate to calculate the force from the predicted position adjustment shortest collision time and collision prediction.

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1 研究背景

近年、映画やゲームといったコンテンツ上で 3 次元コンピュータグラフィクス（以下、3DCG）を用いた様々な表現がなされている。様々な物理現象をコンピュータ上でシミュレーションし、リアルなアニメーションを生成する研究が盛んに行われている [1][2][3][4]。一般的なアニメーション表現手法として既存の 3DCG アニメーション手法に、キーフレームアニメーションやサンプリングアニメーションやプロシージャルアニメーションなどがある [5]。

キーフレームアニメーションとは、任意のフレームごとにオブジェクトの形状や位置や回転などの情報をキーフレームとして登録しアニメーションを生成する手法である。キーフレーム間における様々な情報は、ファンクションカーブというオブジェクトの状態変化を表す区分線形補間を行い計算を行うことでアニメーションを生成する。ファンクションカーブはアニメーションカーブとも呼ばれる。このキーフレームやファンクションカーブやファンクションカーブの補完方法を編集することで、オブジェクトの動きなどのオブジェクトの状態変化を表現しアニメーションを生成する。

サンプリングアニメーションとは、現実の人物や物体の動きを数値化しコンピュータに取り込みコンピュータ上で動きを再現しアニメーションを生成する手法である。サンプリングアニメーションの例としてモーションキャプチャなどがある。サンプリングアニメーションは、現実の人物や物体の動きをリアルタイムで画面上で CG キャラクターなどを動かしたり、運動解析などの様々な研究の分野でも用いられている。サンプリングした数値データが必ずしも正確でない場合があり、その数値データを修正、補完する作業を行う必要がある。

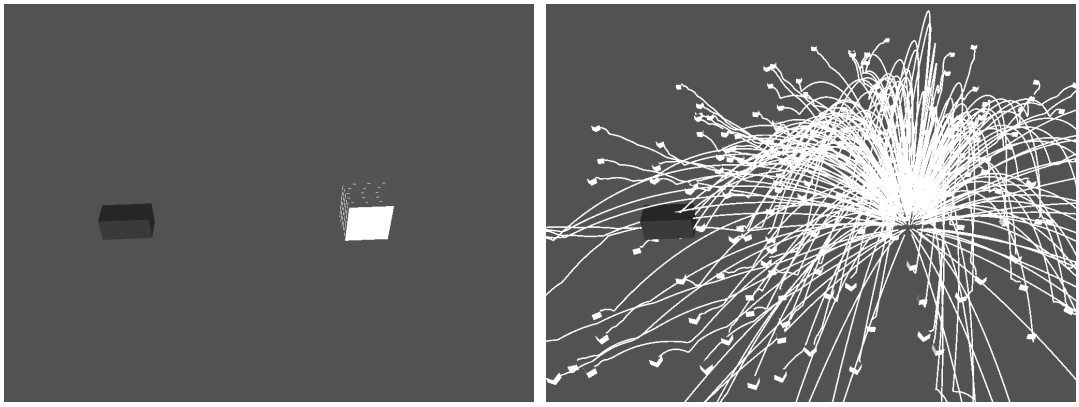
プロシージャルアニメーションとは、物理法則などの数式やパラメータを設定することによって、アニメーションを生成する手法である。プロシージャルアニメーションの例として物理シミュレーションなどがある。キーフレームアニメーションやサンプリングアニメーションに対し、物理法則などの数式やパラメータ

からアニメーションを生成する。オブジェクトの形状や位置や回転の情報を数式や数値データを用いることでアニメーションを生成する。物理法則や数式などをもとにアニメーションを生成しているため、リアリティのあるアニメーションを生成できる。パラメータを変更することで、様々な動きを生成できる。

3DCG において物理シミュレーションは、リアルなアニメーションを生成するために重要な手法の一つである。物理シミュレーションによって、剛体や流体などの挙動を様々なパラメータを設定し計算することでリアリティのある動きを容易に表現することができる [6][7]。

しかしながら、一般に物理シミュレーションは、ユーザーが意図した動きを得るために多くのパラメータの調整が必要になり、扱いづらく出力結果が望ましくないことがある。出力結果が望ましくない例として、物体の爆破シミュレーション時の破片飛散現象表現においてカメラの前で重要なアクションが行われているとき、カメラに対して多くの破片が降り注いでしまい、破片しか映らないシーンができてしまうといったことがある。その結果、重要なアクションが破片によって遮られたり、重要なアクションを見逃すことがある。

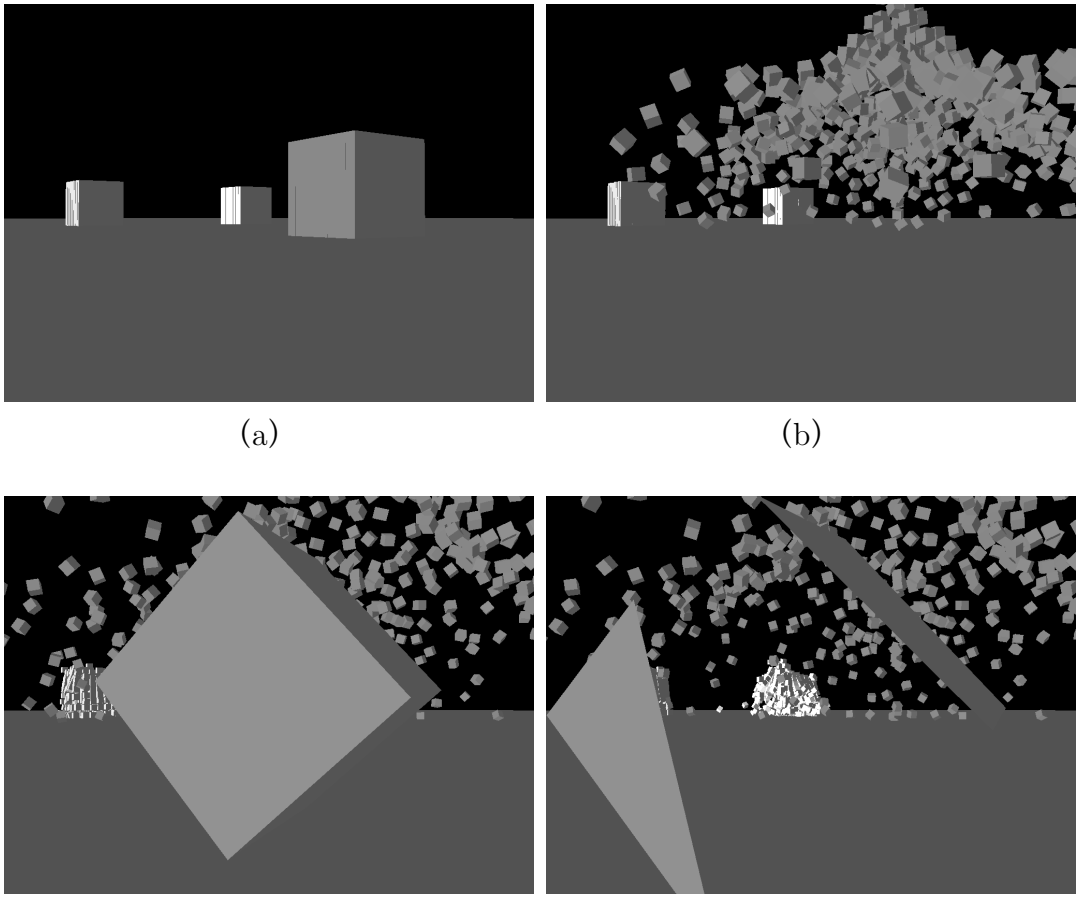
本研究では、この問題に着目し、物体の爆破シミュレーション時の破片飛散現象表現において、破片の挙動を制御するシミュレーション手法を提案する。図 1.1 では、本研究で対象とする物体爆破シミュレーション時の破片飛散現象表現において、出力結果が望ましくないときのカメラと破片の軌跡を示す。黒い直方体がカメラを示し、白い立方体が爆破飛散するオブジェクトを示す。



(a) (b)

図 1.1: 爆発時飛散する破片がカメラ（黒い箱）に当たってしまう例

図 1.1（a）は、シミュレーション開始時の立方体とカメラの様子を示し、図 1.1（b）は、立方体が爆発したあとを示す。図中の白い実線が四散した破片の軌跡を示す。図 1.1 のとき、カメラと飛散した破片が衝突してしまい、モデルの一部が切り取られた不自然な描画結果や破片が突然消えるといった描画結果となり不自然に見えるため望ましくない。



(a) (b)

(c) (d)

図 1.2: 破片がニアクリップ面に衝突し、不自然な描画になっている例

図 1.2 は、(a)、(b)、(c)、(d) と時間が進んでいるものである。図 1.2 は、図 1.2（a）にシミュレーション開始時のモデルの様子を示し、図 1.2（b）のように爆発が起こりカメラに破片が降り注いだ場合の図である。図 1.2（c）において、飛んできた破片モデルがカメラの前を覆ってしまい、画面奥の動きが不明瞭になっている。図 1.2（d）において、立方体の一部が切り取られた不自然な描画結果になっている。

ニアクリップ面とは、視錐台というカメラから見える 3D 空間領域のカメラに最も近い面を指す。視錐台の形状は、平行投影と透視投影の 2 種類投影方法によって形状が異なる。視錐台の形状は、平行投影の場合は直方体になり、透視投影の場合は、四角錐の頭頂点付近を底面と平行に切断した形状をしている。本研究では、

ゲームなどの利用を想定しているため、以降の説明は透視投影の場合の視錐台を用いる。

図 1.2 のようにモデルの一部が切り取られた不自然な描画結果になってしまう原因は、視錐台のニアクリップ面と立方体が干渉してしまうため起こっている。図 1.3 は、図 1.2（d）の状態のときの視錐台のニアクリップ面と立方体が干渉している様子を模式的に示したものである。

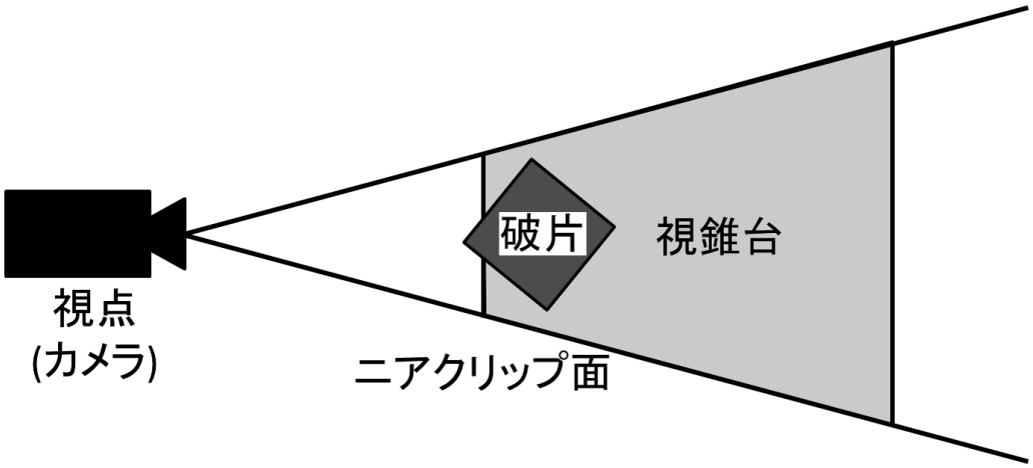


図 1.3: 破片がニアクリップ面に衝突したときの模式図

爆発時飛散する破片は、シミュレーション開始時に破片の行先を予測することはできず、前記のような事態を回避するようにパラメータを設定するのは困難である。

カメラの動きや物体の衝突状況などが予測できない変化を示すゲームシステムでは、キーフレームアニメーションなどの手付けアニメーションやサンプリングアニメーションでは対応できない。また、このようなゲームシステムでは、パラメータの調整だけではユーザーが意図した破片の挙動を得ることが困難である。

物理シミュレーションのようなプロシージャルアニメーションによりユーザーが望んだ結果を得るためには、多くのパラメータ調整に試行錯誤が必要である。それらのパラメータの調整を行い、もっともらしく望ましいアニメーション結果を得るための研究が多数行われている [8][9][10][11]。

Twigg ら [12] は、さまざまな物理パラメータを変えて複数回のシミュレーションを行い、出力結果の軌跡からユーザーが目的とする結果を取捨選択する手法を提案した。Popović, ら [13] は、ユーザーが物体の挙動を対話的に変更することができる手法を提案した。ユーザーが指定した挙動に合うような初期パラメーターを再計算し、ユーザーは画面に表示される結果を直接変更することで目的とする出力結果を得ることができる。近藤ら [14] や三谷ら [15] は、対話的に流体、弾性体のアニメーション制御を行った。Chenney ら [16] は、拘束条件を満たす複数のアニメーションを生成し、事前に定義したもっともらしさの評価基準により最良の結果を選ぶ手法を提案した。Twigg ら [17] は、多数の剛体モデルが望ましい配置になるように時間ステップを逆に計算する手法を提案した。また、川田ら [18] は、爆発後の煙粒子が望みの形状になるようステップごとに煙粒子を制御しつつ物理特性も保つ手法を提案した。これらの手法は、複数候補を生成することなく望ましい結果に導くことができる利点がある。

今給黎ら [19] は、破壊シミュレーション後の形状を制御する手法を提案した。この手法でもユーザーは事前に破壊対象となる形状のばねモデルの強弱を指定する必要があり、目標とするカメラ位置に対応した破片の動きにこの手法を応用することは困難である。

対話的にパラメータを決定する Popović, ら [13] や近藤ら [14] や三谷ら [15] の手法では、ユーザーが指定した挙動に合うような初期パラメーターを再計算している。ユーザーが目指す出力結果となるようなパラメータを取捨選択する Twigg ら [12] や Chenney ら [16] の手法では、さまざまな物理パラメータを変えて複数回のシミュレーションを行い、ユーザーが目的とする結果を取捨選択することができる。しかしながら、これらの手法では繰り返し処理や結果の再計算処理や事前に望ましい状況を指定するが必要である。ゲームではそのような手順を踏むことは許容できない。

様々な物理現象を時間ステップを逆にする Twigg ら [17] や川田ら [18] の手法では、複数候補を生成することなく望ましい結果に導くことができる利点がある。し

かしながら、ユーザーが、望ましい状況を固定的な形として直接指定する必要があり、状況に応じた多数の剛体モデルの制御に応用することは難しい。

このように、カメラの動きや物体の衝突状況などが予測できない変化を示すゲームシステムでは、多くの剛体破片を望ましい動きに導くことは難しい。ユーザーがどのような操作を行うかは分からず、動的であるカメラの位置や爆発時飛散する破片の動きは一意的に決まらず、パラメータの制御は極めて困難である。無理に動きを制御すると、破片の衝突によって不自然な動きが目立ってしまう結果となる。

本研究では爆破時四散する破片の挙動に注目した。本研究では、カメラと破片の衝突予測位置から最小限の力を加える事で、ニアクリップ面との干渉しないように破片の挙動を制御を行った。カメラの最大移動速度からカメラと破片が衝突する可能性のある衝突候補となる破片を検出する。衝突候補の放物運動の放物線とカメラの最大移動範囲となる球と交差判定を行い、最短衝突予測時刻を求め、衝突候補が投影面上を通るときの予測位置を検出する。その後、最短衝突予測時刻と衝突予測位置から調整力を計算し衝突候補となる破片に加える。

その結果、物理シミュレーションのこの問題に対して動的なカメラ操作に対応し、違和感なくカメラを避ける破片のリアルタイム剛体シミュレーション手法を実現した。そして、提案手法をプログラムで実装し、検証を行い提案手法の有用性を確認した。

1.2 論文構成

本研究では、動的なカメラ操作に対応し、違和感なくカメラを避ける破片のリアルタイム剛体シミュレーション手法を実現することを目的とし研究を行った。

論文の構成は以下の通りである。第 2 章では、本研究で提案する破片飛散現象表現におけるカメラ回避制御手法について述べる。第 3 章では、本研究で目標とする動きに対して、単純な方法でカメラと破片との衝突回避を行った場合の問題点と研究対象について述べる。第 4 章では、本研究で開発した破片飛散現象表現

におけるカメラ回避制御手法のプログラムにより、その結果の検証と考察を行う。

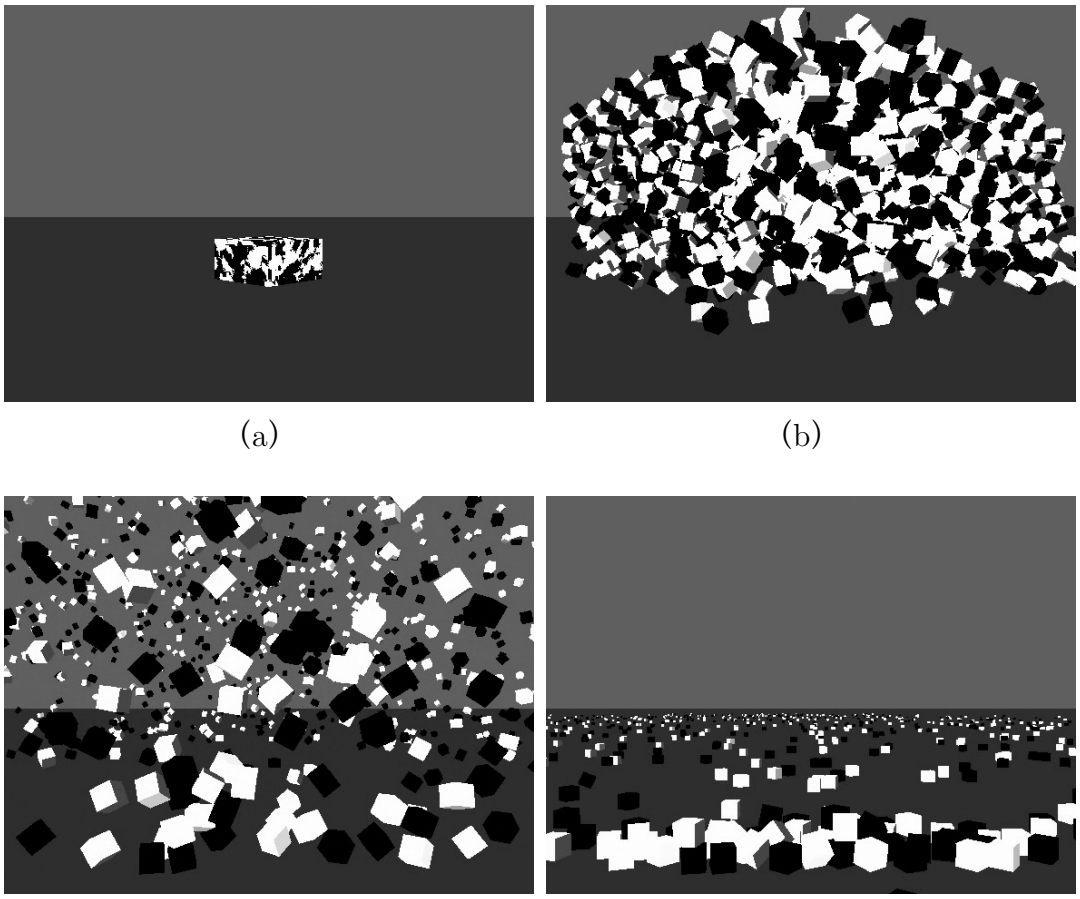
第5章では、本研究の成果と意義をまとめ、今後の展望について述べる。

第 2 章

単純なカメラと破片との衝突回避手法のその問題点

2.1　単純なカメラと破片との衝突回避手法

図 2.1 は、破片モデルがカメラやニアクリップ面に衝突しないように、カメラの前に透明な壁を設置したものである。



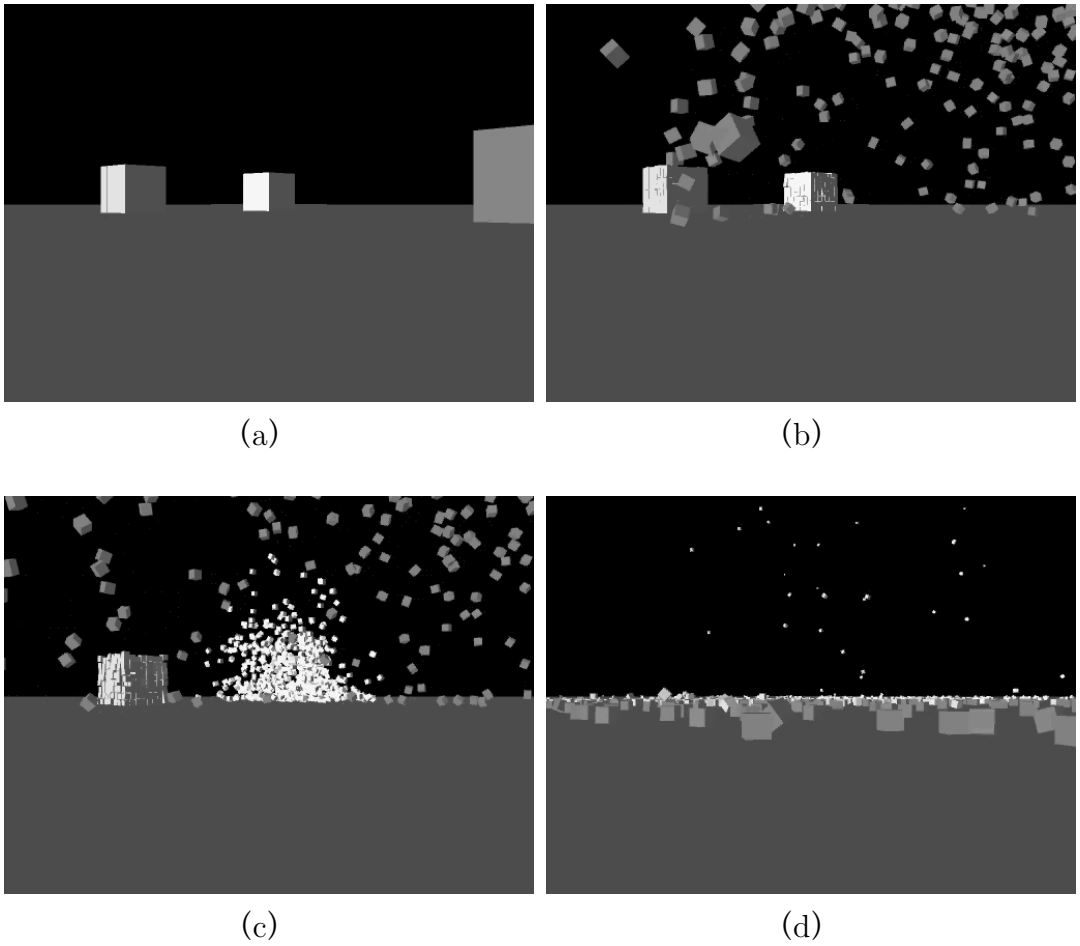
(a)　(b)

(c)　(d)

図 2.1: カメラの前に透明な壁を設置した場合

図 2.1 は、(a)、(b)、(c)、(d) と時間が進んでいる。図 2.1 (a) にシミュレーション開始時のモデルの様子を示し、図 2.1 (b) のように爆発が起こりカメラに破片が降り注いだ場合の図である。図 2.1 (c) において、飛んできた破片モデルがカメラの前を覆ってしまい、画面奥の動きが不明瞭になっている。図 2.1 (d) ではカメラの前に破片が集まってしまって不自然な形になってしまう。また透明な壁と飛散した破片が当たった場合、カメラと破片があったったかのような跳ね返りが起こってしまい、望ましくない。

また図2.2は、図1.2と同じパラメータを用い、カメラに飛んで来るすべての破片に対して力を加え、カメラから破片を逸らした例である。



(a) (b)

(c) (d)

図2.2: カメラに飛んでくすべての破片を逸らした例

図2.2は、(a)、(b)、(c)、(d) と時間が進んでいる。図2.2 (a) にシミュレーション開始時のモデルの様子を示し、図2.2 (b) のように爆発が起こり、カメラに降り注ぐ破片に力を加えた場合の図である。図2.2 (c) において、カメラの前を覆う破片が減り、画面奥の動きが明瞭である。図2.2 (d) では、カメラと破片は衝突せず、オブジェクトの一部が切り取られたような不自然な描画結果は起こっていない。しかし図2.2 (d) では、カメラの周囲に破片が降り注がず、迫力を損なうため望ましくない。同時に破片は力が加わったかのような不自然な破片挙動になる。

図 2.1、図 2.2 のような方法では破片の挙動に違和感が生じてしまう。本研究では、見た目では破片に力が加わっているか区別がつかないように、必要最小限の力を加える。そうすることで、破片は本来の軌跡から大きくずれるような不自然な軌道をとらない。また、映画やアニメなどでカメラに破片が飛んでくる場合、画面の上下左右方向に抜けていくことがある。これにより、破片が突然消えることやカメラの周囲に破片が降り注がないといったことにならず、不自然な破片挙動にならず迫力を損なわない。図 2.3 は、奥から手前に向かって破片が実線のような軌道をとっている様子を示す。この図は映画などで画面の上下左右方向に抜けていく様子を模式的に示したものである。

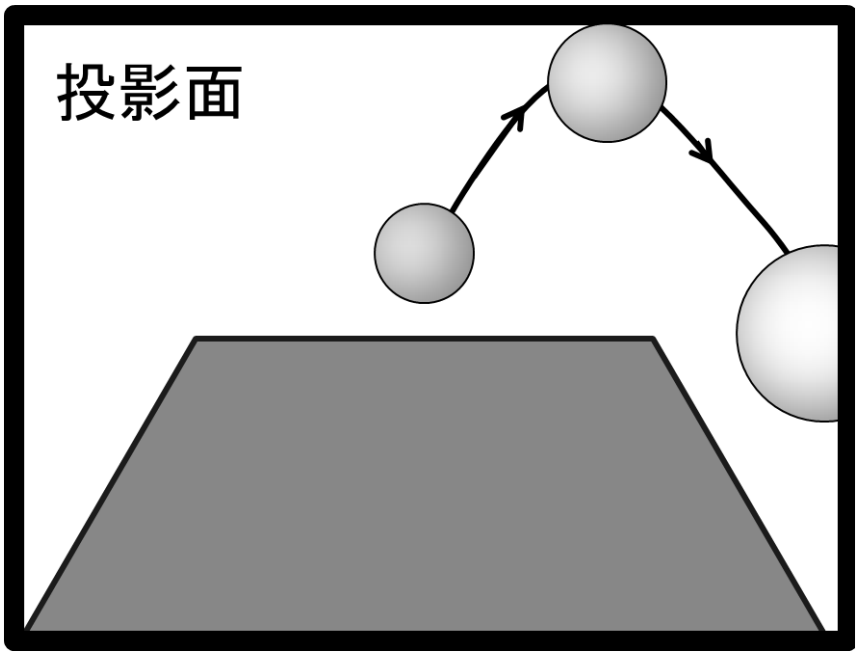


図 2.3: 破片が画面の上下左右方向に抜けていく様子

カメラが「見る」ことができる領域はビューボリュームといい、ビューボリュームの形は視錐台という [20][21][22][23]。ビューボリュームは視体積とも呼ばれる。ビューボリュームの形状は、投影方法により形状が決まる。投影とは次元を減らすことを指す。3 次元の物体を 2 次元であるディスプレイ上で表現する投影が必要

である。投影方法には、平行投影と透視投影の 2 種類がある。平行投影の場合は視錐台は直方体になり、透視投影の場合は、四角錐の頭頂点付近を底面と平行に切断した形状をしている。図 2.4 は、透視投影の場合の視錐台の形状を示す。図中の灰色の領域が視錐台の領域である。

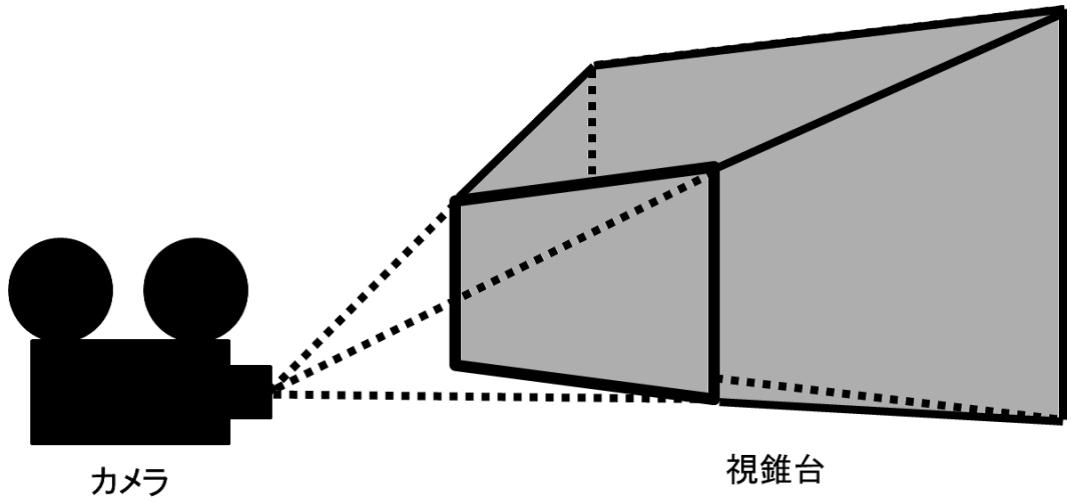


図 2.4: 透視投影の場合の視錐台

図 2.5 は、透視射影変換における視錐台の各部位の名称を示した図である。

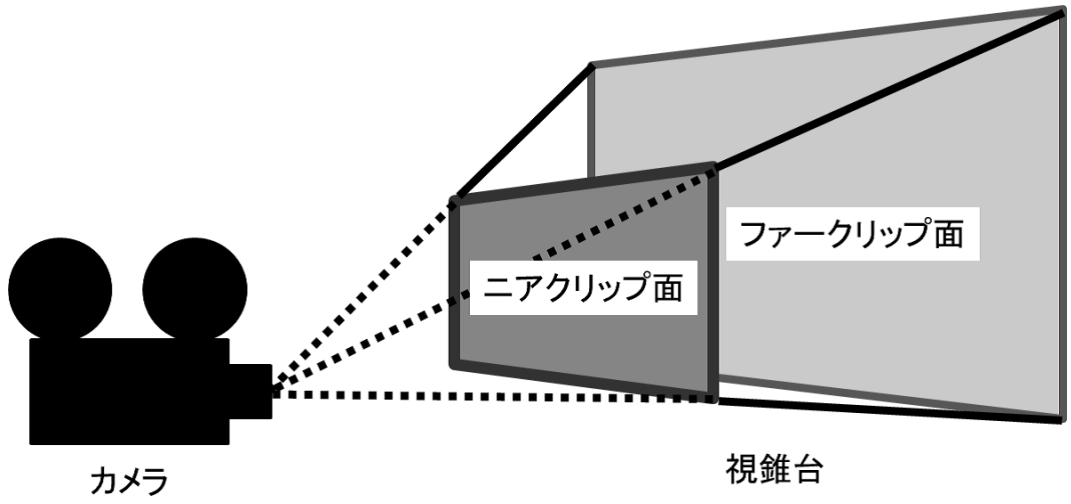


図 2.5: 視錐台の各面の名称

視錐台の形状はカメラを頂上とする四角錐の先端部を切り取った形状をし、クリップ面という 6 つの面で表される。カメラ位置に一番近い視錐台の面を、ニアクリップ面といい、カメラ位置に一番遠いの面を、ファークリップ面といい、それ以外の面をそれぞれ上、下、左、右クリップ面という。ニアクリップ面は、オブジェクトにカメラが近づき過ぎないようにすることが望ましいため設定する。ファークリップ面は、カメラが描画できる距離を制限するために設定する。カメラから見えるオブジェクトの数はニアクリップ面からファークリップ面までの距離に比例する。

破片がカメラに衝突することを防ぐ方法として、ニアクリップ面をカメラに近づけるといった方法がある。しかし、カメラからニアクリップ面が近すぎる場合、描画精度が落ちて z ファイティングと呼ばれるノイズが起きる。図 2.6 は、z ファイティングが起こった様子を示す。図中の黒い 4 角形と白い 4 角形の z バッファの値が同じとなり面が重なり、重なった面にノイズが起きる。

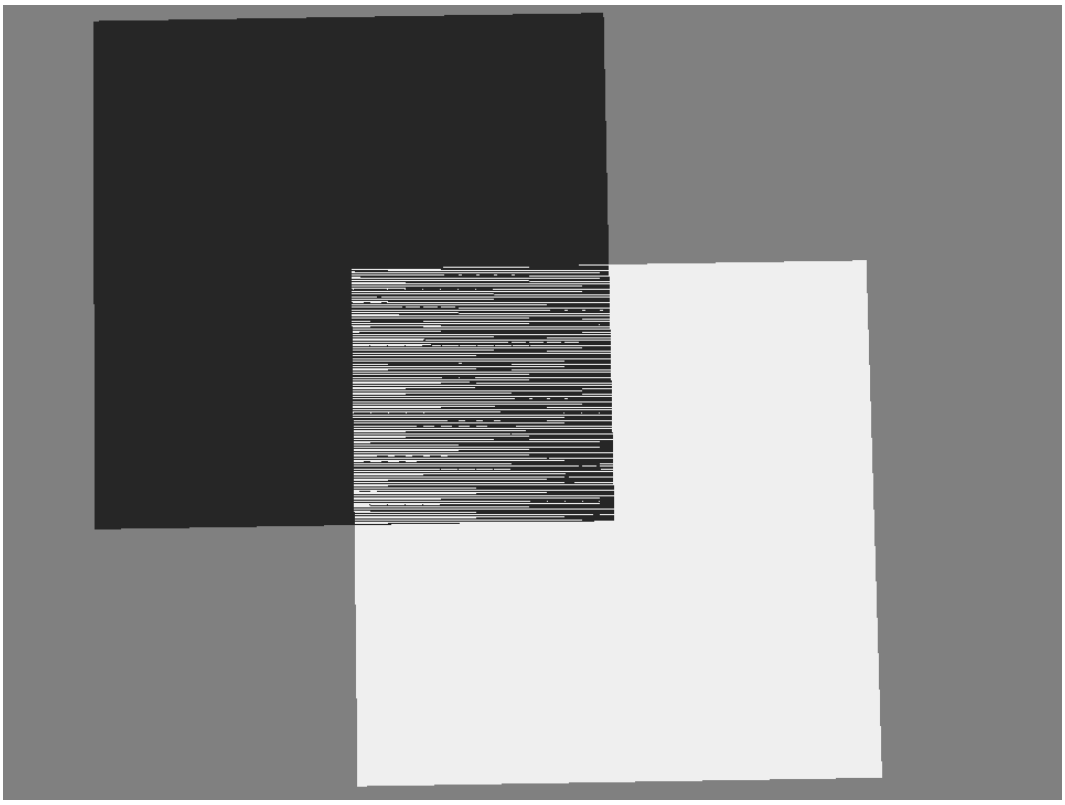


図 2.6: z ファイティングが起こった様子

2.2 研究目標

本手法はゲーム等のインタラクティブなコンテンツで用いることを想定する。動的なカメラや事前に予測できないタイミングで爆発飛散する破片の動きに対応する必要がある。

本研究で目標とする爆発時飛散する破片の軌跡は、次の 3 点を満たすものとする。

- 動的なカメラの動きに対応している。
- 破片の軌道を変える際、破片の動きの違和感をなくすため、カメラの上下左右方向に破片を逸らす。
- カメラの前を覆うような破片が飛んくるときやカメラに対して破片が飛んでくるとき、物理的に多少不自然でもカメラから破片を逸らすことを優先する。

本研究では、カメラに降り注ぐ破片に対して力を加えることで破片の動きを制御し、カメラから違和感なく逸らすような破片の軌跡を描くリアルタイム剛体シミュレーション手法を提案する。破片が動くカメラに当たらずに、破片はカメラの上下左右方向に逸れるような軌道をとることを目標とする。図 2.7 は、爆破直後はカメラに向かう破片が、最終的には破片がカメラの上下左右方向に逸れるような軌道をとる様子を模式的に示したものである。実線はカメラと破片が衝突してしまう場合の軌跡を示し、点線が本研究で目標とする破片の軌跡を示したものである。

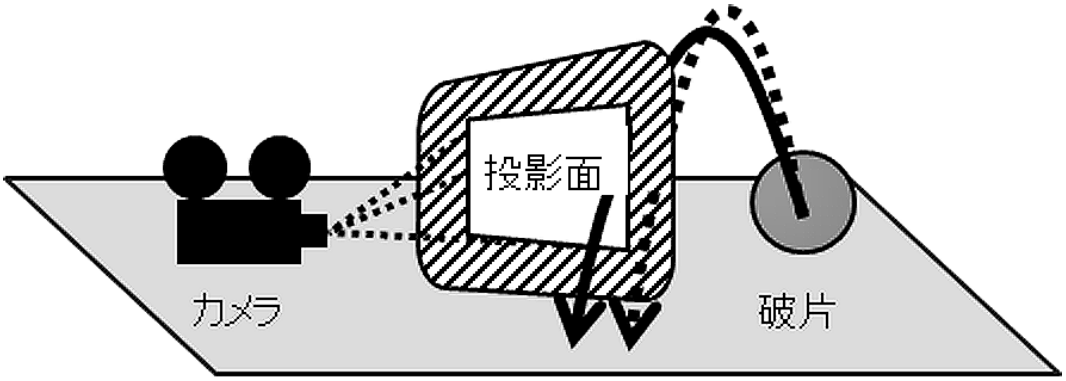


図 2.7: 本研究で理想とする破片の軌道

第 3 章

提案手法

3.1　提案法の概要

図 3.1 は、本章での手法の概略を示した図である。

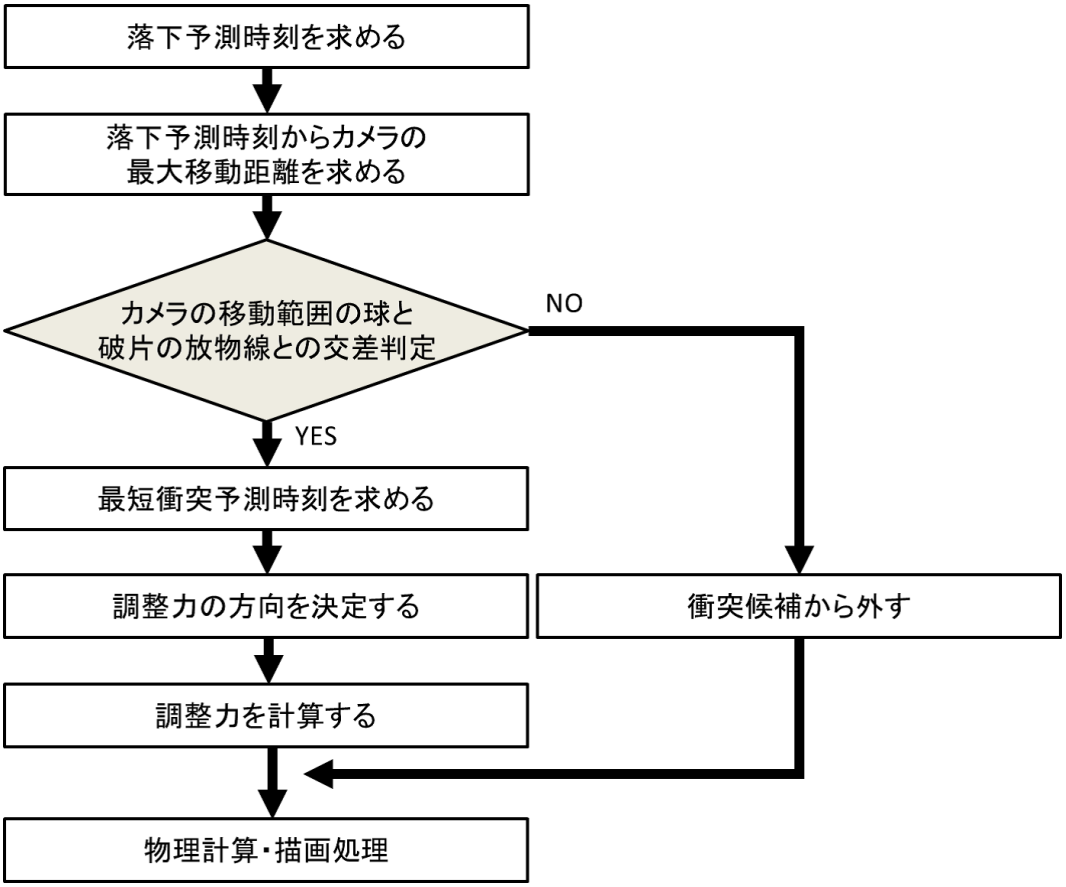


図 3.1: 提案手法の流れ

はじめに衝突予測計算を行い、衝突する可能性のある衝突候補を絞り込む手法について、3.2 節で述べる。次に、衝突候補に加える力の向きを決める手法について、3.3 節で述べる。最後に衝突候補に加える力の大きさを決定する手法について、3.4 節で述べる。

3.2 衝突予想計算

爆破直後の破片が地面に落ちるまでの時間にカメラが移動可能な領域を考え、その領域と破片の予想軌跡との交差判定を行うことで、カメラに当たる可能性がある破片を検出する。提案法ではカメラの移動最大速度 $\mathbf{V}_c$ は決まっているものとする。爆破時の時刻を $t = 0$ とする。時刻 t における破片 p の位置を $\mathbf{P}_p(t) \equiv (x_p(t), y_p(t), z_p(t))$、速度を $\mathbf{V}_p(t) \equiv (V_p^x(t), V_p^y(t), V_p^z(t))$ とすると、破片 p が描く放物線の軌跡は次の式によって表される。

$$\begin{aligned} x_p(t) =& x_p(0) + V_p^x(0)t \\ y_p(t) =& y_p(0) + V_p^y(0)t - \frac{1}{2}gt^2 \\ z_p(t) =& z_p(0) + V_p^z(0)t. \end{aligned} \tag{3.1}$$

ただし g は重力加速度の値である。

このとき破片 p が地面に落ちる予想時刻 t_p^G は、式 (3.1) で y をカメラの y 座標とおいて t についての二次方程式を解く事で求まる。このとき予想時刻までにカメラが動く最大距離 $R(t)$ は次の式によって表される。

$$R(t) = V_c(t_p^G - t) \tag{3.2}$$

時刻 t におけるカメラ位置を $\mathbf{P}_c(t) \equiv (x_c(t), y_c(t), z_c(t))$ とすると、爆破時点 $(t = 0)$ から時刻 t までにカメラが動きうる領域は次の式で定義できる球となる。

$$|\mathbf{P}_c(t) - \mathbf{P}_c(0)|^2 = R(0)^2. \tag{3.3}$$

式 (3.1) と式 (3.3) を連立させると、t に関する 4 次方程式となる。これを解いて t の範囲が $[0, t_p^G]$ となる解が存在すれば、破片 p はカメラと衝突する可能性があると $t = 0$ の時点で推定できる。この時の t の解 t_p^C を最短衝突予想時刻と呼ぶ事にする。この一連の計算を $t = 0$ における衝突予想計算と呼ぶ事にする。また、このようにしてカメラと衝突する可能性があると判定された破片を衝突候補と呼ぶ事にする。

図 3.2 は、衝突予想計算により衝突の可能性が予想され、最短衝突予想時刻が求められる様子を示す。このときの破片 p は、次のタイムステップでも衝突候補となる。

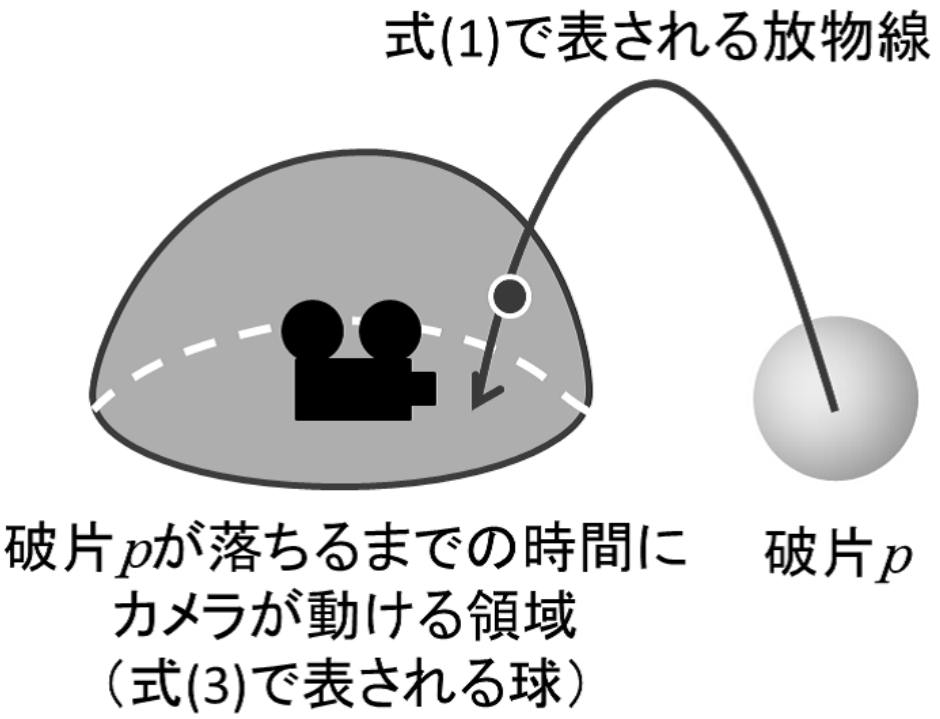


図 3.2: 衝突予想計算

3.3 衝突候補に加える力の向きの決定

爆破直後の衝突候補を求めるには、飛散するすべての破片について $t = 0$ における衝突予想計算を行う。理論的には、爆破後に放物線を描く破片の初速度は、爆破に伴う力の作用が終わった時点での速度とすべきであるが、提案法での計算では、$t = 0$ の瞬間で爆破力の作用が終わり、放物運動の初速度が求められる。そのため、式 (3.1) における $\mathbf{V}_p(0)$ は物理シミュレーションにより得られる値を使う。

$t = 0$ における衝突予想計算で衝突候補とならなかった破片は、以降（$t > 0$）の各タイムステップでの衝突予想計算は行わない。タイムステップ n での時刻 $t = t_n > 0$

における衝突予想計算は $t = 0$ のものと同様である。破片の放物線を示す式 (3.1) とカメラが動きうる範囲を示す式 (3.3) についても、$t = 0$ を $t = t_n$ に置き換えた式を用いる。

加える力の向きは、衝突候補が投影面上を通るときの予測位置を検出し、投影面の四辺（上下左右）のうちもっとも近い辺に向かう垂線の向きとする。図 3.3 は、衝突候補に加える力の向きの決定ときの様子を示す。

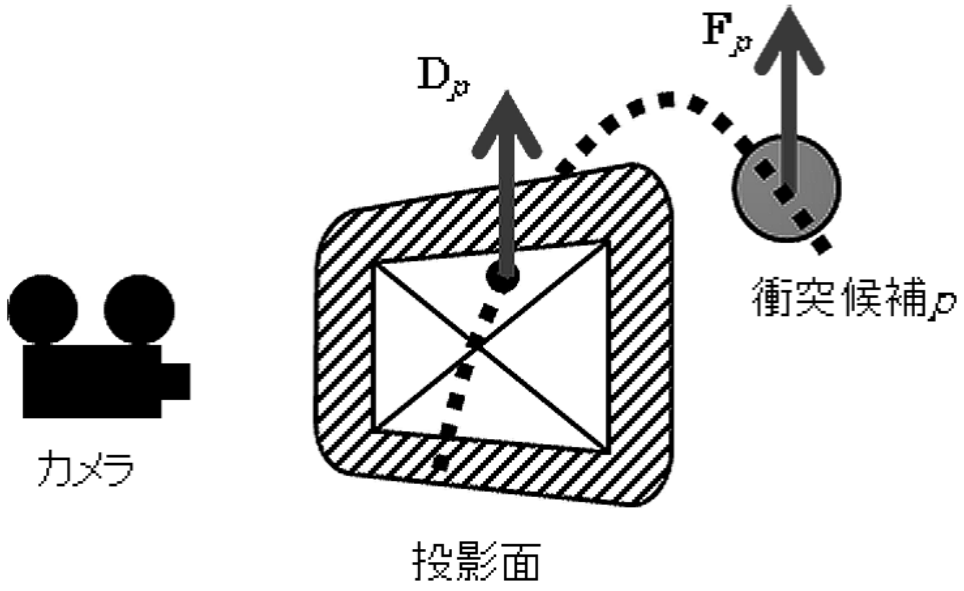


図 3.3: 衝突候補に加える力の向きの決定

3.4 衝突候補に加える力の大きさの計算

飛散した破片のうち、衝突候補に対しては意図的に力を加え、図 2.7 で斜線で示した領域、すなわち投影面の枠の外側に飛んでいくように制御することが望ましい。このとき加える力が大きすぎると不自然な動きになるため、最小限の力で画面外へ衝突候補を逸らす必要がある。

カメラから遠い衝突候補 p は、大きな力を加えなくとも、小さな力で大きく衝

突起動から逸らすことができる。逆に、カメラから近い衝突候補 p は、小さな力では衝突してしまう恐れがあるため大きな力を加える必要がある。そのため、カメラから遠い衝突候補 p には、小さな力を加え、カメラから近い衝突候補 p には、大きな力を加える。同様に最短衝突予想時刻まで余裕がある衝突候補 p については小さな力を、逆に最短衝突予想時刻が迫っている衝突候補 p については大きな力を加えるというものである。

衝突候補 p に加える力を $\mathbf{F}_p$、破片 p の投影面位置からもっとも近い辺への垂線に対応する世界座標での向き（単位ベクトル）を $\mathbf{D}_p$、カメラに対する破片 p の相対速度を $\mathbf{V}_{pc}$、カメラに対する破片 p の相対距離を $\mathbf{P}_{pc}$、t は現在時刻 p の最短衝突予想時刻を t_p^C、k はユーザー設定の比例定数、とすると次の式が成り立つ。衝突候補 p が衝突候補から外れるまで $\mathbf{F}_p$ を毎フレーム算出し加え続ける。

$$\mathbf{F}_p = k\frac{|\mathbf{V}_{pc}|}{|\mathbf{P}_{pc}|^2(t_p^C - t)}\mathbf{D}_p \tag{3.4}$$

式 (3.4) は、カメラに対する破片 p の相対速度の比例式である。また、現在時刻 t と最短衝突予想時刻を t_p^C から求められる破片 p の衝突まで時間とカメラに対する破片 p の相対距離を $\mathbf{P}_{pc}$ による反比例の式である。カメラに対する破片 p の相対速度が早く衝突するまで余裕が無い衝突候補 p には、確実にカメラから逸らすために強い力を加えることができる。カメラにと破片 p との距離が離れていれば、衝突までの余裕があるとみなし、小さな力を加える。また、現在時刻 t と最短衝突予想時刻を t_p^C から求められる破片 p の衝突まで時間に余裕がある場合は、余裕があるため小さい力を加える。

第 4 章

検証と考察

4.1 実行結果

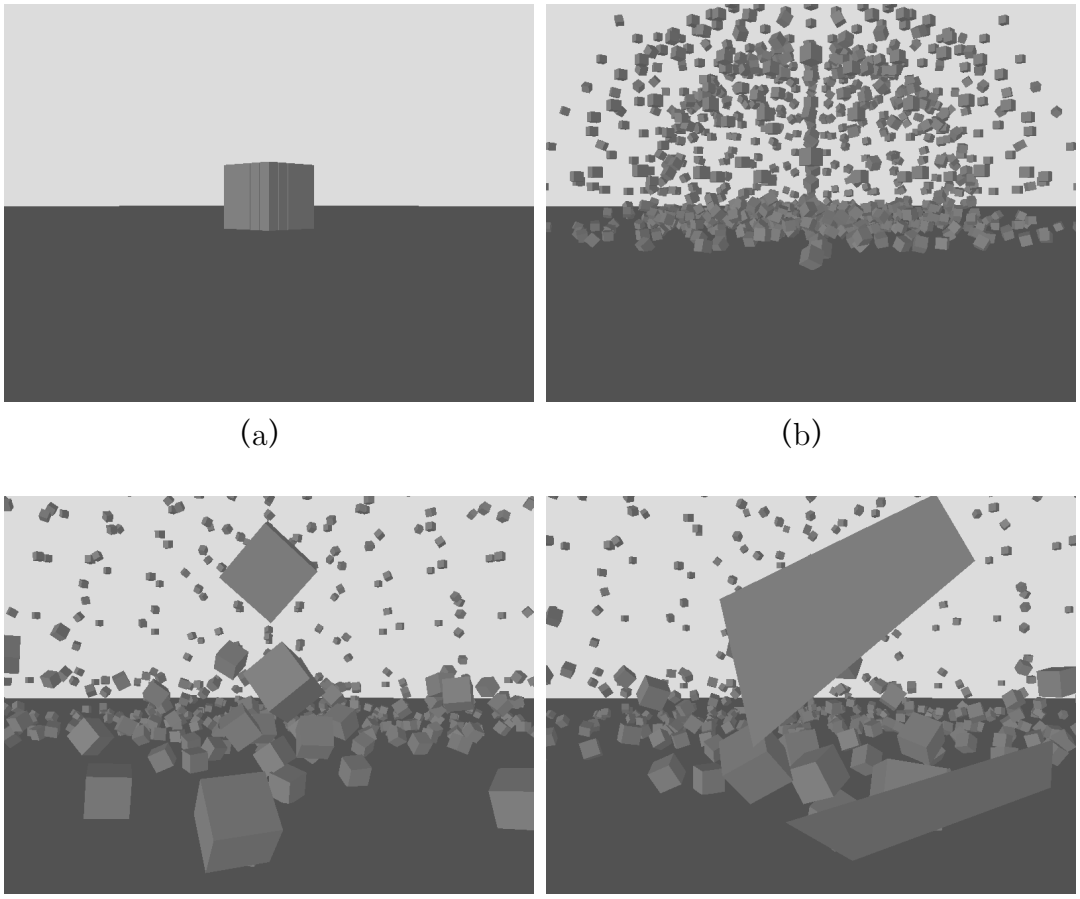
提案手法を物理演算エンジンライブラリである NVIDIA 社 [24]PhysX[25] を用いて実装し、本手法の有用性を検証した。検証に用いたプログラムは、グラフィックス API の OpenGL[26] をベースとした 3 次元グラフィックスツールキットである Fine Kernel Tool Kit System[27] を用いて実装した。検証に使用した環境を次の表 4.1 に示す。

表 4.1: 実行環境

OS	Windows 7 64bit
CPU	Intel®Core™i7-2630QM CPU @ 2.00GHz
GPU	NVIDIA GeForce GT 525M
ビデオメモリ	2724 MB
メインメモリ	4GB

本シミュレーション実験では、カメラの正面にある立方体が爆発して更に小さい立方体に分かれて飛散する。カメラは正面に向かって移動させている。立方体は X 方向、Y 方向、Z 方向に対して 13 分割し、最大で 1728 個の立方体に分かれるものとしている。本手法ではゲームなどのインタラクティブコンテンツでの利用を想定している。一般的なゲームの描画速度は 1/60 秒であるため、シミュレーションの 1 ステップの時間は 1/60 秒とする。1/60 秒よりも長く設定してしまうと、映像が滑らかでなくなってしまう。

図 4.1 は、提案手法を用いず、爆発時の飛散する破片シミュレーションを行った場合の実行結果である。



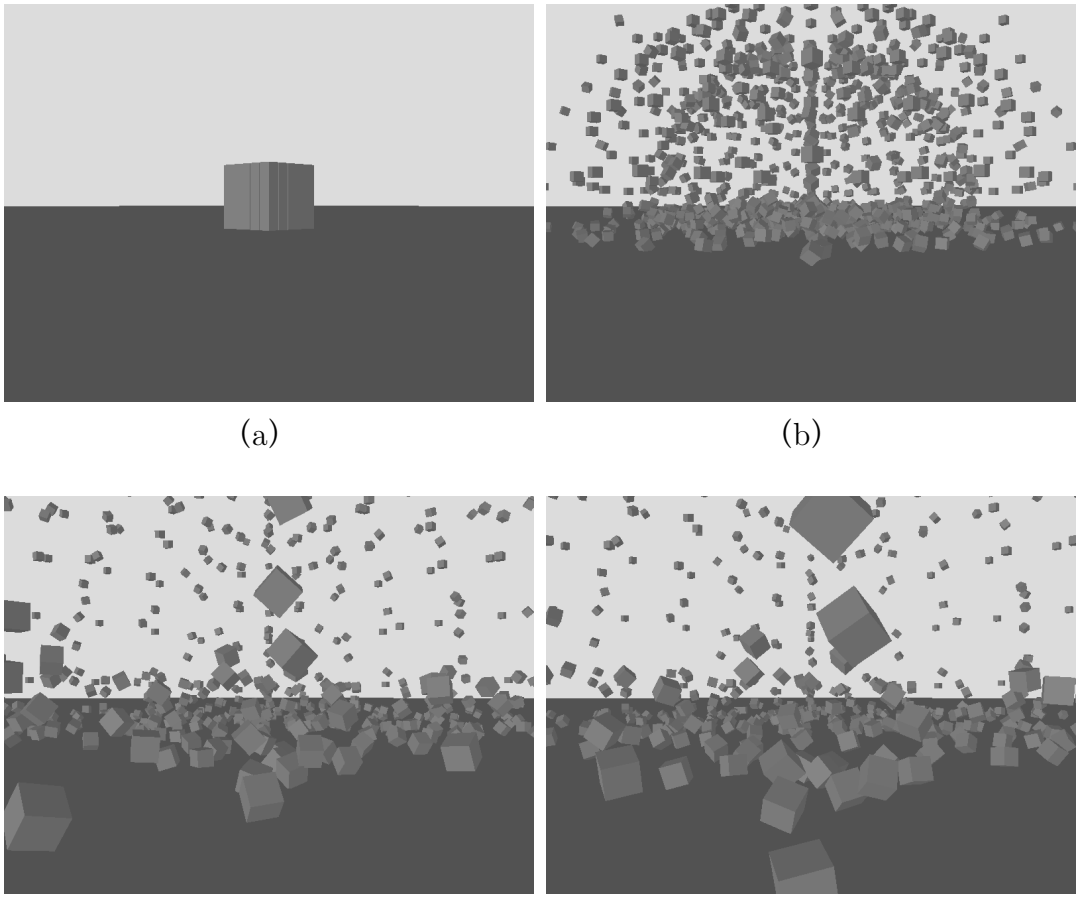
(a) (b)

(c) (d)

図 4.1: 本手法を用いない場合の破片の飛散する様子

図 4.1 は、図 4.1（a）にシミュレーション開始時のモデルの様子を示し、図 4.1（b）のように爆発が起こりカメラに破片が降り注いだ場合の図である。図 4.1（c）では、飛んできた破片モデルがカメラの前を覆ってしまい、画面奥の動きが不明瞭になり、望ましくない描画結果になっている。図 4.1（d）では、ニアクリップ面に当たっているため画面左上の破片に大きな穴があいたような望ましくない描画結果になっているのがわかる。

図 4.2 は、提案手法を用いて、爆発時の飛散する破片シミュレーションを行った場合の実行結果である。



(a) (b)

(c) (d)

図 4.2: 本手法を用いた場合の破片の飛散する様子

図 4.2 は、図 4.2（a）にシミュレーション開始時のモデルの様子を示し、図 4.2（b）のように爆発が起こりカメラに破片が降り注いだ場合の図である。図 4.2（c）では、飛んできた破片モデルがカメラの前を覆わずに、画面奥の動きが明瞭になっている。図 4.1（d）では、ニアクリップ面に当たっているため画面左上の破片に大きな穴があいたような望ましくない描画結果になっていた。しかし、図 4.2（d）では、破片とカメラが衝突せずに、カメラの上下左右方向に抜けていくような破片挙動になった。

図 4.1、図 4.2 は、いずれも（a）、（b）、（c）、（d）と時間が進んでいるものである。

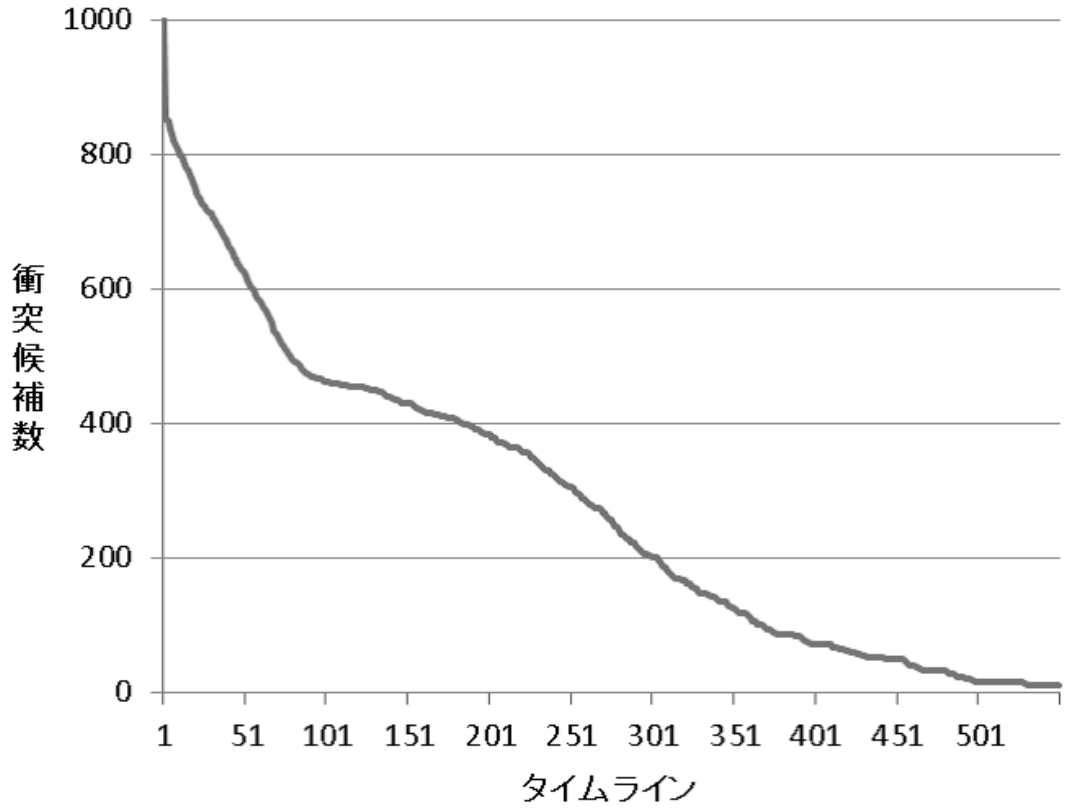
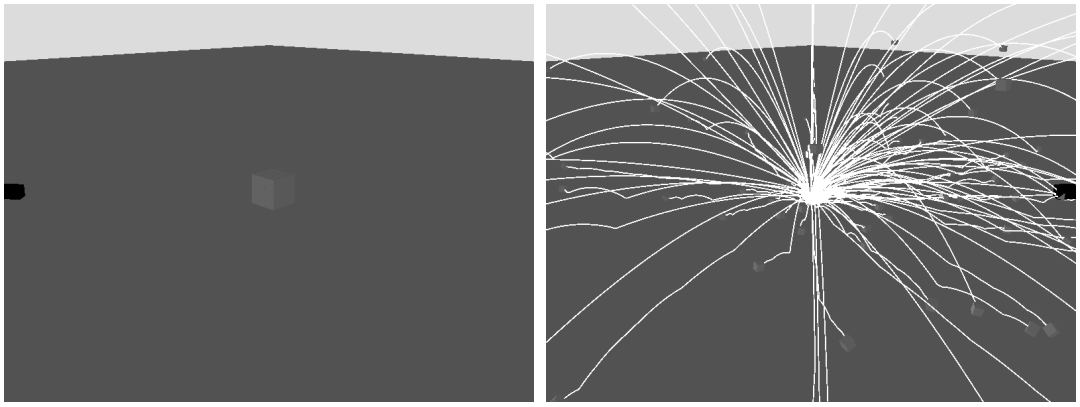


図 4.3: タイムステップと衝突候補数との関係

図 4.3 は、カメラの正面にある立方体が爆発して更に立方体は 1000 個の立方体に分かれて飛散するシミュレーションをおこなった結果である。各タイムステップごとに衝突予想計算を繰り返し行うことで、毎ステップごとに衝突候補は少しずつ減少する。

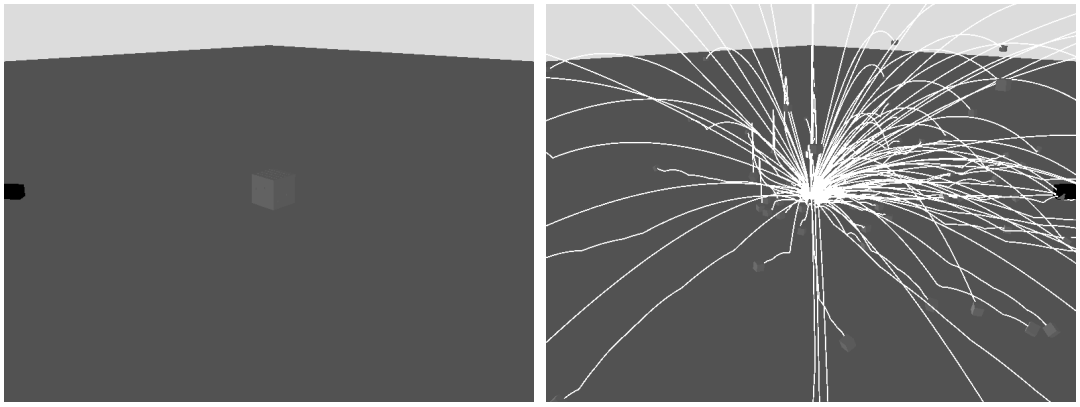
図 4.4 は、提案手法を用いず、爆発時の飛散する破片シミュレーションを行った場合の各破片の軌跡を示したものである。図 4.4 は、カメラの正面にある立方体が爆発して更に立方体は 125 個の立方体に分かれて飛散するシミュレーションをおこなった。このとき破片とカメラが衝突している。



(a) (b)

図 4.4: 本手法を用いずに破片飛散した時の軌跡

図 4.5 は、提案手法を用いて、爆発時の飛散する破片シミュレーションを行った場合の各破片の軌跡を示したものである。このとき破片とカメラが衝突は起こっていない。



(a) (b)

図 4.5: 本手法を用いて破片飛散した時の軌跡

図 4.4（a)、図 4.5（a）では黒い直方体がカメラを示し、白い線が破片の軌跡を示す。図 4.4（a)、図 4.5（a）は、シミュレーション開始時のモデルの様子を示し、図 4.4（b)、図 4.5（b）のように爆発が起こりカメラに破片が降り注いだ場合の図である。

表 4.2 は、本手法を用いた場合の 1 秒間当たりの描画回数を表したものである。なお、描画速度の単位は frames per second（以下「fps」） である。

表 4.2: 描画速度の測定結果

オブジェクト数	描画速度（fps）	
	本手法を用いない場合	本手法を使った場合
1000	125.00	111.11
1331	90.91	76.92
1728	71.43	47.61

表 4.3 は、衝突候補を用いない場合と衝突候補を用いた場合の描画速度を表したものである。

表 4.3: 衝突候補を用いた場合の測定結果

オブジェクト数	描画速度（fps）	
	衝突候補を用いない場合	衝突候補を使った場合
1000	83.33	111.11
1331	50.03	76.92
1728	29.66	47.61

4.2 考察

本研究で目標とする爆発時飛散する破片の軌跡は 2.2 節で述べた、次の 3 点を満たすものとする。

- 動的なカメラの動きに対応している。
- 破片の軌道を変える際、破片の動きの違和感をなくすため、カメラの上下左右方向に破片を逸らす。
- カメラの前を覆うような破片が飛んくるときやカメラに対して破片が飛んでくるとき、物理的に多少不自然でもカメラから破片を逸らすこと優先する。

また、本手法はゲーム等のインタラクティブなコンテンツで用いることを想定すためリアルタイム性が必要である。

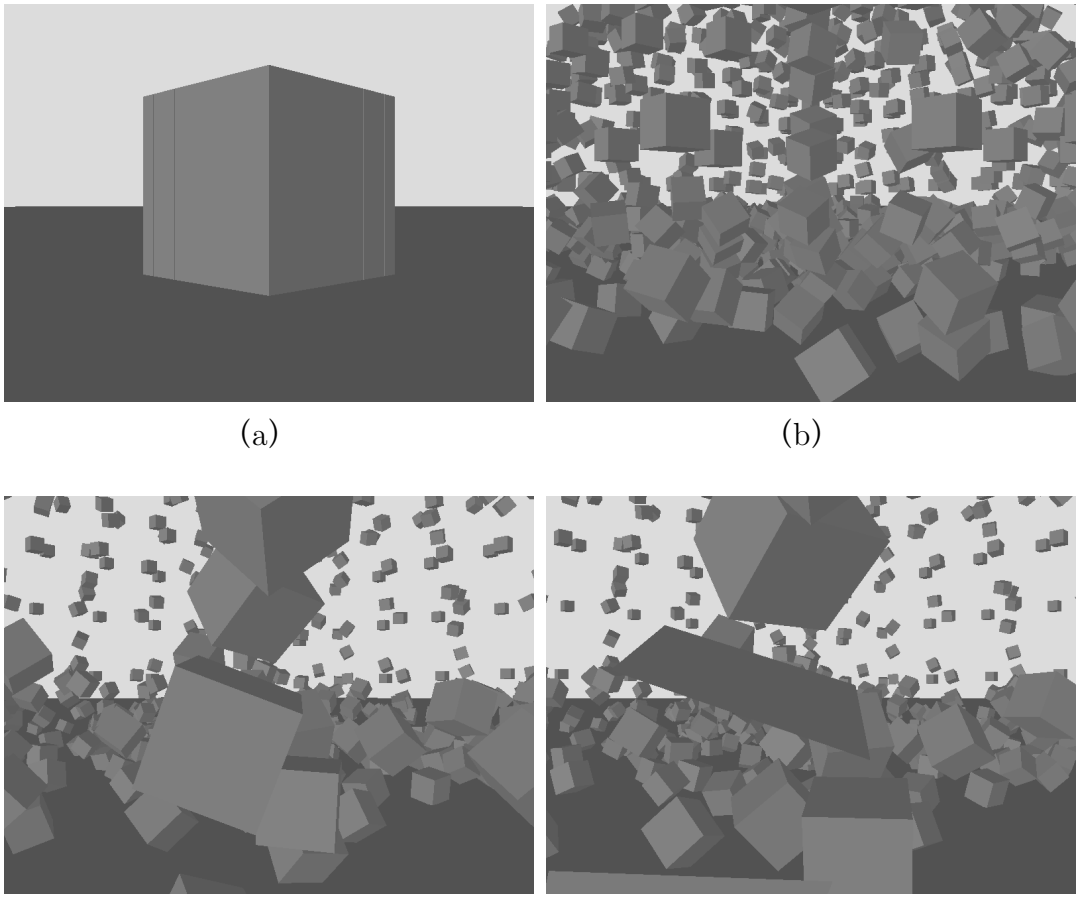
図 4.1（c）では、飛んできた破片モデルがカメラの前を覆ってしまい、画面奥の動きが不明瞭である。しかし、図 4.2（c）では、破片が画面を遮らず、画面奥の動きが明瞭である。

また図 4.1（d）では、ニアクリップ面に当たっているため画面左上の破片に大きな穴があいている。しかし、図 4.2（c）では、飛散した破片はカメラに当たらずに画面の上下左右方向に逸れている。また、図 4.5 は図 4.4 と比較して、一部の破片だけが本来の軌跡からズレている。

表 4.2 と表 4.3 に示すように、破片の数が 1331 個であった場合であっても 60fps 以上の描画速度がある。破片の数が 1728 個であった場合でも 30fps 以上の描画速度がある。1 秒間に 30〜60fps 以上の描画速度を維持できている。このため、毎フレーム破片の軌道を逸らすための計算を行なってもリアルタイム性を維持することが可能である。

4.3 現状の問題点

本手法の課題は、カメラの至近距離で四散表現が起こる場合、破片とカメラが衝突してしまい、望ましくない描画結果になってしまう。図 4.6 は、カメラの至近距離で四散表現が起こり、望ましくない描画結果である。



(a) (b)

(c) (d)

図 4.6: 至近距離で四散現象が起こったときの様子

図 4.6 は、(a)、(b)、(c)、(d) と時間が進んでいるものである。図 4.6 は、図 4.6（a）にシミュレーション開始時のモデルの様子を示し、図 4.6（b）のようにカメラの至近距離で爆発が起こり、カメラに破片が降り注いだ場合の図である。図 4.6（d）において、立方体の一部が切り取られた不自然な描画結果になっている。本手法では、破片の動きの違和感をなくすため最小限の力を破片に加えている。そのため、図 4.6 の状況では、破片がカメラから逸れる軌道をとるまでに十分な距離がないため破片がカメラと衝突してしまう。

この問題の解決方法として、第 3.4 節の式 (3.1) で用いているユーザー設定の比例定数 k の値を大きくすることで、この現象を回避できる。しかし、カメラから離れた位置での四散表現がある場合に、破片の動きが大きく変化し破片の動きに

違和感が生じる。破片に対して過度に力が加わることでカメラ付近に降り注ぐ破片が減り、臨場感が失われる描画結果となる場合がある。また、本来の軌跡から大きくずれるような不自然な破片の挙動をする結果となる可能性がある。

第 5 章

おわりに

本研究では、動的なカメラに向かって飛んでくる破片に対して単純な力を加えるのではなく、逸らす位置も重要と考え、逸らす位置を考慮したリアルタイム剛体シミュレーション手法を提案した。カメラの動きに対応して破片の軌道を変える際、動きの違和感をなくすため、上下左右のうち最も適切な方向に破片を逸らす。さらに、衝突の危険性が高いほど強い力で破片の軌道変更制御を行うようにした。これにより、物理的に多少不自然でもカメラに破片が当たらないことを優先するという動きが実現できた。

本手法の課題は、カメラの至近距離で四散表現が起こる場合、破片とカメラが衝突してしまい、望ましくない描画結果になってしまう。この問題の解決方法として、ユーザー設定の比例定数 k の値を大きくすることで、この現象を回避できる。しかし、カメラから離れた位置での四散表現がある場合に、破片の動きが大きく変化し破片の動きに違和感が生じる。破片に対して過度に力が加わることで臨場感が失われる描画結果となったり、本来の軌跡から大きくずれるような不自然な破片の挙動をする結果となる可能性がある。

本手法では並列計算などの高速化処理を行なっていない。本手法を並列計算を行うことで、対応できる破片の数を増やすことができる。各フレーム毎にすべての衝突候補を計算するのではなく、衝突候補をいくつかのグループに分け、各フレームごとに別のグループを計算することで衝突予測計算を高速化できる。このとき衝突の危険性のある衝突候補を優先的に計算することで、カメラと衝突候補との衝突を防ぐことができる。

今後、破片がカメラから逸れて行くだけでなく、破片が飛んでくるまでの”タメ”の表現が可能になればと考えている。また、破片がカメラの上下左右のうちどの辺に多く現れるかの割合や飛んでくる頻度を、ユーザーの意図に応じて調整可能になればと考えている。これらが実現できれば、様々な演出面において、本研究の有用性が向上する。

なお、本研究は情報処理学会グラフィクスと CAD 研究会第 148 回研究発表会において“ 3DCG におけるカメラ視点を考慮したリアルタイム剛体アニメーション

に関する研究”[28] として発表した内容を含む。

謝辞

本研究を締めくくるにあたり、研究の指針及び開発の手法、論文の執筆と幅広いご指導ご教授を頂きました、主査であり本校メディア学部の渡辺大地講師、並びに副査の三上浩司講師、柿本正憲教授に厚く感謝いたします。そして、様々な相談に乗ってくださった研究室のすべての皆様に感謝いたします。

参考文献

[1] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 154–159. Eurographics Association, 2003.

[2] B.M. Klingner, B.E. Feldman, N. Chentanez, and J.F. O'brien. Fluid animation with dynamic meshes. In *ACM Transactions on Graphics (TOG)*, Vol. 25, pp. 820–825. ACM, 2006.

[3] 原田隆宏, 田中正幸, 越塚誠一, 河口洋一郎. GPU を用いたリアルタイム剛体シミュレーション (セッション 5: CG 処理の高速化, テーマ: CG と記録及び CG 一般). 情報処理学会研究報告. グラフィクスと CAD 研究会報告, Vol. 2007, No. 13, pp. 79–84, 2007.

[4] F. Liu, T. Harada, Y. Lee, and Y.J. Kim. Real-time collision culling of a million bodies on graphics processing units. In *ACM Transactions on Graphics (TOG)*, Vol. 29, p. 154. ACM, 2010.

[5] 画像情報教育振興協会. ディジタル映像表現 改訂版. CG ー ARTS 協会, 改訂, 11 2006.

[6] R.D. Henderson. Scalable fluid simulation in linear time on shared memory multiprocessors. In *Proceedings of the Digital Production Symposium*, pp. 43–52. ACM, 2012.

[7] D.M. Kaufman, T. Edmunds, and D.K. Pai. Fast frictional dynamics for rigid bodies. In *ACM Transactions on Graphics (TOG)*, Vol. 24, pp. 946–956. ACM, 2005.

[8] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. In *ACM Transactions on Graphics (TOG)*, Vol. 22, pp. 716–723. ACM, 2003.

[9] R. Fattal and D. Lischinski. Target-driven smoke animation. In *ACM Transactions on Graphics (TOG)*, Vol. 23, pp. 441–448. ACM, 2004.

[10] 朱暁宇, 伊藤弘樹, 菊池司. ユーザ制御可能な風による樹木の揺れのビジュアルシミュレーション. 第 26 回 Nicograph 論文コンテスト (2010 年 Nicograph 秋季大会) 概要集, CD-ROM, 2010.

[11] S. Sato, Y. Dobashi, T. Yamamoto, and K. Anjyo. Controlling simulated explosions by optimization and prediction. In *Computer-Aided Design and Computer Graphics (CAD/Graphics), 2011 12th International Conference on*, pp. 296–301. IEEE, 2011.

[12] C.D. Twigg and D.L. James. Many-worlds browsing for control of multibody dynamics. In *ACM Transactions on Graphics (TOG)*, Vol. 26, p. 14. ACM, 2007.

[13] J. Popović, S.M. Seitz, M. Erdmann, Z. Popović, and A. Witkin. Interactive manipulation of rigid body simulations. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 209–217. ACM Press/Addison-Wesley Publishing Co., 2000.

[14] 近藤亮, 金井崇, 安生健一. 物理指向補間による弾性体アニメーションの制御手法. Visual Computing／グラフィクスと CAD 合同シンポジウム, pp. 5–10, 2006.

[15] 三谷純, 五十嵐健夫. 流体シミュレーションを統合した対話的な形状設計手法. *Workshop on Interactive Systems and Software*, pp. 25–28, 2008.

[16] S. Chenney and D.A. Forsyth. Sampling plausible solutions to multi-body constraint problems. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 219–228. ACM Press/Addison-Wesley Publishing Co., 2000.

[17] C.D. Twigg and D.L. James. Backward steps in rigid body simulation. *ACM Transactions on Graphics (TOG)*, Vol. 27, No. 3, p. 25, 2008.

[18] 川田玄一, 金井崇. 物理モデルを考慮した爆発のモデリング. Visual Computing ／グラフィクスと CAD 合同シンポジウム, Vol. 20, , 2011.

[19] 今給黎隆, ジョハンヘンリー, 西田友是. 破壊後の形状の制御を伴う破壊シミュレーション. 画像電子学会誌, Vol. 38, No. 4, pp. 449–458, 2009.

[20] 魏大名. コンピュータグラフィックス (IT Text). オーム社, 12 2003.

[21] Fletcher Dunn and Ian Parberry. 実例で学ぶゲーム 3D 数学. オライリージャパン, 10 2008.

[22] James D. Foley, Steven K. Feiner, Andries van Dam, and John F. Hughes. コンピュータグラフィックス 理論と実践. オーム社, 3 2001.

[23] ジェイソン・グレゴリー. ゲームエンジン・アーキテクチャ (Professional game programming). ソフトバンククリエイティブ, 12 2010.

[24] NVIDIA Corporation. NVIDIA Developer Zone. `https://developer.nvidia.com/`.

[25] NVIDIA Corporation. — NVIDIA Developer Zone. `https://developer.nvidia.com/technologies/physx`.

[26] Silicon Graphics International Corp. OpenGL. `http://www.opengl.org/`.

[27] 渡辺 大地. Fine Kernel Tool Kit System. `http://www.teu.ac.jp/media/~earth/FK/`.

[28] 成田晃, 渡辺大地, 三上浩司, 柿本正憲, 竹内亮太. 3DCG におけるカメラ視点を考慮したリアルタイム剛体アニメーションに関する研究. 情報処理学会研究報告, Vol. 148, No. 9, 2012.