修士論文　　2023 年度

# 水面とエネルギー波の相互作用に関する研究

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

蘇 路文

修士論文　　2023 年度

# 水面とエネルギー波の相互作用に関する研究

指導教員　渡辺 大地 教授

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

蘇 路文

# 論 文 の 要 旨

| | |
|---|---|
| 論文題目 | 水面とエネルギー波の相互作用<br>に関する研究 |
| 執筆者氏名 | 蘇 路文 |
| 指導教員 | 渡辺 大地 教授 |
| キーワード | 流体シミュレーション、One-way カップリング、<br>レイ・キャスティング、GPU、リアルタイム |


[**要旨**]

エネルギー波が水面を通過して波紋を引き起こす現象はアニメーションによく見られる。そして、ゲームなどのリアルタイムシステムにおける現在の実装は、主にレンダリングまたは事前計算されたアニメーションという形になっている。グラフィックスにおける物理シミュレーションの研究成果が数多くある一方で、このような現実には存在しない現象を物理シミュレーションという形で実装する研究は少ない。同時に、現在の物理シミュレーションのほとんどは、ゲームのようなリアルタイム環境に直接適用できるほど効率的ではない。

本研究では、シミュレートするための水面と剛体との One-Way フレームワークを提案する。このフレームワークの目的は、一般的なシミュレーション方法に比べて、レイトレーシングやスペースクリッピングのようなレンダリング技術を組み合わせて計算負荷を減らすことである。また、エネルギー波と水面の相互作用するためのユーザー定義パラメーターも提供する。そのための計算プロセスを提案し、簡単な物理モデルを用いた試みを行う。一方、この計算フローを簡略化するために、CPU と GPU ベースの最適化アイデアをそれぞれ提案する。これにより、通常、リアルタイムシステムにおいて事前に計算されたアニメーションの実装を必要とする現象を、物理シミュレーション的に実現することができる。その結果、この研究は許容可能な外観を達成し、実験証明によれば、最適化された方法は、効果に影響を与えることなく、計算する頂点の数を減らすことに成功した。しかし、単純な物理流体モデルを使用したため、システムは一般的なケースに対して十分に安定していなかった。

# A b s t r a c t

| | |
|---|---|
| Title | A study on the interaction<br>between water surfaces and energy |
| Author | Luwen Su |
| Advisor | Taichi Watanabe |
| Key Words | Fluid Simulation, One-way Coupling<br>Ray Casting, GPU, Real Time |

[**summary**]

The phenomenon of energy waves passing through water surfaces and causing ripples is common in animation. And current implementations in real-time systems such as games are primarily in the form of rendered or pre-computed rendered or pre-computed animations. The reason for this, however, is that while there are many research results on physics simulation in graphics, However, there is little research on the implementation of such phenomena that do not exist in reality in the form of physical simulations. At the same time, most current physics simulations are not efficient enough to be directly applicable to real-time environments such as games. In this study, we propose a One-Way with water surface and rigid body to simulate framework is proposed. The purpose of this framework is to compare it to common simulation methods, such as ray tracing and rendering techniques such as space clipping in combination to reduce the computational load. In addition, a user-defined parameter for the interaction of energy waves and water surface and user-defined parameters for the interaction of energy waves and water surfaces. A computational process for this purpose is proposed and attempted using a simple physical model. On the other hand, CPU- and GPU-based optimization ideas are proposed to simplify this computational flow, respectively. This allows phenomena that normally require the implementation of pre-computed animations in real-time systems, be realized in a physical simulation manner. As a result, the study achieves an acceptable appearance, and according to experimental proofs, the optimized method, successfully reduced the number of vertices to be computed without affecting the effect. However, because a simple physical fluid model was used, the system was not sufficiently stable for the general case.

# 目　次

# 図 目 次

# 第 1 章

# はじめに

## 1.1 研究の背景と目的

アニメーションや漫画では、衝撃波が水面を押し広げ、水面に波紋を作り出すシーンがよく描かれる。しかし、このような現象をリアルタイムの物理シミュレーションで正確に表現することは大変難しい。この再現が難しい主な理由は、以下の点に由来する。

物理シミュレーションの分野では、精密で自然に近いシミュレーション結果を得るため、通常、高度な計算手法を駆使して各粒子ごとに計算を行う。映画制作などのオフライン環境では、事前計算といった手法で満足のいく結果を得られる。しかし、60FPS を要求するゲーム環境のようにリアルタイム性が求められる場面では、計算リソースが主にレンダリング、モデルのロード、AI、システム通信に割り当てられており、物理シミュレーションに充てられる時間や能力は限定される。家庭用のシステムの限界から、これらの不足を補うために更に計算リソースを割くわけにはいかない。その結果、リアリズムを追求するよりも、効率を優先したシンプルかつ効率的な物理シミュレーションアルゴリズムの使用が必要になる。もし特定のアルゴリズムが効率性の観点から 60FPS での運用が難しい場合、そのアルゴリズムの採用は見送られる。

ゲーム内の流体には、速度に対する要求は高いが、リアリズムに対する要求はそれほど高くないという特性がある。このため、標準的な物理シミュレーション手法を直接ゲームに適用するのは難しい。さらに、リアルタイムの物理シミュレーション手法も、通常はゲーム内の流体表現には向かないことが多い。代わりに、事前に計算されたレンダリング手法やアニメーション化されたレンダリング手法が用いられることが一般的である。例えば、コップの中の水や小瓶の中の液体、水の魔法などの小規模な流体シミュレーションにおいては、オフラインレンダリングを通じて実現し、アニメーションは事前に制作してゲーム内でループする。大規模な水面の場合は、それを平面とみなすことができる。ゲーム内の水はテクスチャとして扱い、法線マッピングや変位マッピングといったポストプロセス技術を使用して、マテリアルが水のように見えるようにする。このアプローチの利点は、GPU 処理を活用するため非常に高速であり、リアルタイム環境で使用可能である点である。ただし、周囲の環境との相互作用がないというデメリットがある。

図 1.1 は、GPU GEMS 2 Chapter 18. Using Vertex Texture Displacement for Realistic Water Rendering[1] の技術で実現する水面の例となる。

最近の AI 技術の進歩により、AI が流体方程式の制約条件を自動的に処理して流体シミュ

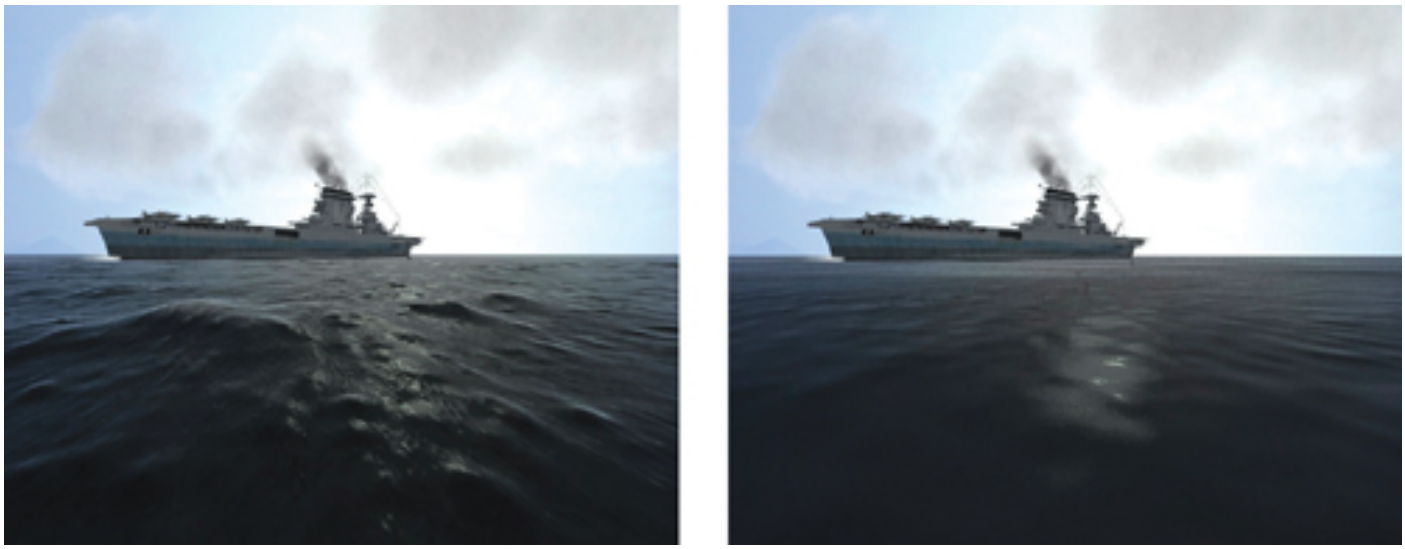


図 1.1: Displacement 技術で実現する水面の例

レーションを行う手法が発展している。Raissi, M. ら [2] は Physics-informed neural networks（PINN）モデルを提案したが、リアルタイムシミュレーションに適用するためには、まだ克服すべき問題がある。3D 技術の発展以降、よりリアリスティックなレンダリングや、アクションを鮮明にするキャラクターモーションシミュレーションなど、没入感とゲーム性を兼ね備えた仮想世界の実現に向けた取り組みが進められてきた。しかし、計算能力やアルゴリズムの限界から、物理シミュレーションはまだ簡易な剛体シミュレーションに留まっている。衝撃波などの現象は、アーティストが作成したアニメーションをゲームでループする技術が主流である。一方で、リアルタイムシミュレーションが難しい水の現象は、レンダリング技術で表現することが多い。本研究の目的は、アーティストによるアニメーションに頼らず、ゲームアプリケーション向けの物理シミュレーションによって、レンダリングが難しい現象をゲーム内で実現する方法を探求することである。本研究では、ゲームエンジンで一般的に使われるエネルギー波の生成技術を用いて、エネルギー波を定義する。物理シミュレーションを用いて、現実には存在しないエネルギー波が水面を通過し波紋を生み出す現象の計算プロセスを定義する。陽解法、大きなステップサイズ、計算の簡易性を活かした SWE[3]（Shallow Wave Equation、浅い波の方程式）により水面をシミュレートする。リアルタイムレンダリングがオフラインレンダリングに近ずくようにし、エネルギー波の計算が必要な頂点を簡略化する。水面をテクスチャーとしてカリングする。SWE を

使った水面シミュレーションは Kass[3] による手法を用いた。この方法は実装が容易で速いという利点があるが、結果の精度は十分ではない。現在、この方法は主にリアルタイムでの水面シミュレーションが必要ながら、精度がそれほど求められない分野で使用されている。その後、Wang らは CPU と GPU の両方で動作し、より高い安定性とスケーラビリティを持つ General Shallow Wave Equations[4]（GSWE）モデルを提案した。

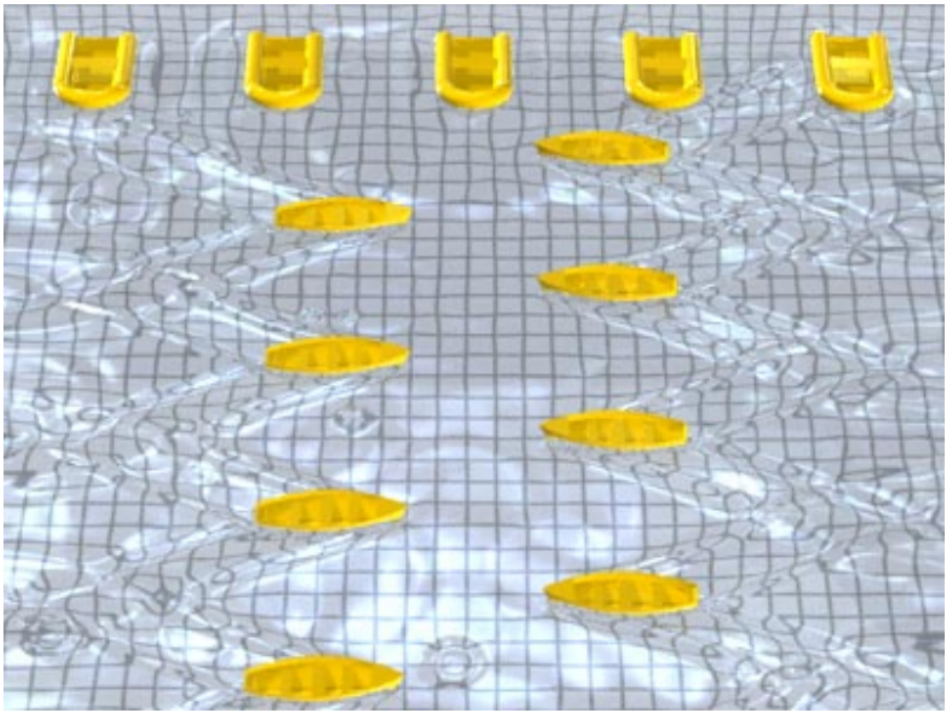


図 1.2: 流体と剛体組み合わせるシミュレーションの例

図 1.2 は Wang ら提案した GSWE 手法の論文 Solving general shallow wave equations on surfaces[4] の中に流体と剛体組み合わせるシミュレーションの例である。

流体と剛体を組み合わせるシミュレーションは、カップリングシミュレーションとも呼ばれ、流体と剛体の相互作用をモデル化する手法である。たとえば、水中を航行する小舟がその例である。流体と剛体のハイブリッドシミュレーションでは、通常、最初に流体シミュレーション用の手法を選定する。その後、剛体が水中に存在する際の速度と、水中で沈んでいる部分が押し出す水の量が水面に与える影響を評価する。そして、双方向の連成シミュレーションでは、剛体が水面に与える影響だけでなく、水面が剛体に与える影響も考慮に入る。さらに、水中における剛体に作

用する浮力も算出する。

本研究では、エネルギー波が水面を横切る現象を、流体と剛体の基本的な連成シミュレーションとして抽象化し、エネルギー波を剛体扱いしながらも、エネルギー波への水面の影響を考慮しないアプローチを採用している。

シミュレーションの最適化は、数学的な理論とプロジェクトの実践の 2 つの側面から考察される。理論面では、リアルタイムレンダリングとオフラインレンダリングの関係を踏まえ、数学モデルの近似や省略、時間ステップの調整を通じて、効率と品質のバランスを見つける。

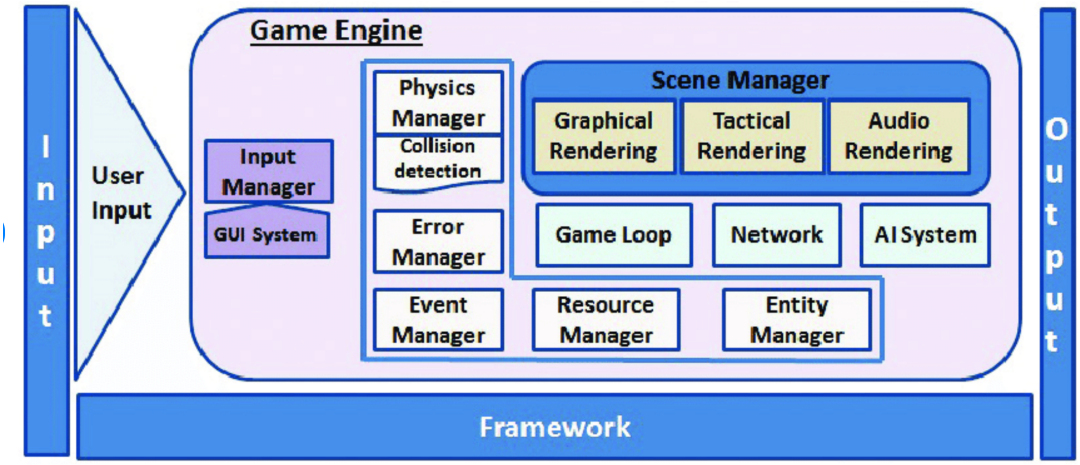



図 1.3: ゲームエンジンで Frame の計算流れの例

一方、工学的な側面からは、必ずしも必要でない頂点の削減などのアプローチが採用される。ゲームエンジンの視点では、1 フレームの計算がレンダリングと物理シミュレーションの二つステップを必要とするため、レンダリングデータの物理シミュレーションプロセスでの再利用も、最適化の一つの方向性として検討される。

図 1.3 は、Anis Zarrad の Game Engine Solutions Chapter 5[5] のゲームエンジンでのフレームの計算フローの例を示している。

本研究では、物理モデルの単純化とプロジェクト側からの最適化の 2 つのアプローチを用いて、水面とエネルギー波のシミュレーションを実施する。

本研究の実験的なアプローチにより、計算する頂点の数を減らしながらも、許容できる画像品質を実現することが可能である。さらに、本実験ではゲームエンジンのスクリプトレイヤーを活用

しており、効率化の余地が存在する。グラフィカル API を用いて、すべての計算プロセスを一つのカスタムパイプラインに統合することは、検討すべき一つの方向である。陽解法の不安定性を考慮すると、より高い効率を達成しつつ、他の安定した水面シミュレーション手法を探究することが必要である。波紋のエッジでラグランジアンを使用して水滴をシミュレートすることは、視覚的な表現を向上させるための有効な手法である。

## 1.2 論文構成

第一章では、本研究の背景および目的を説明する。第二章では、本研究の提案手法を述べる。第三章では、本研究の提案手法のパフォーマンスとロバスト性実験を示す。第四章では、まとめを示す。

# 第 2 章

# 提案手法

## 2.1　エネルギー波

エネルギー波にはさまざまな定義があり、阿部ら [6][7][8] はレンダリングベースのアプローチでエネルギー波を実現する提案した。

エネルギー波は主に 2 つの要素から構成されている。形状は、複数の頂点を組み合わせて作られるモデル。次はテクスチャー (texture), 最終的にどのように見えるかを決める形状とテクスチャー合成し、アニメーションのような、事前設定方式により、繰り返し送る。


図 2.1: エネルギー波のテクスチャーの作る例

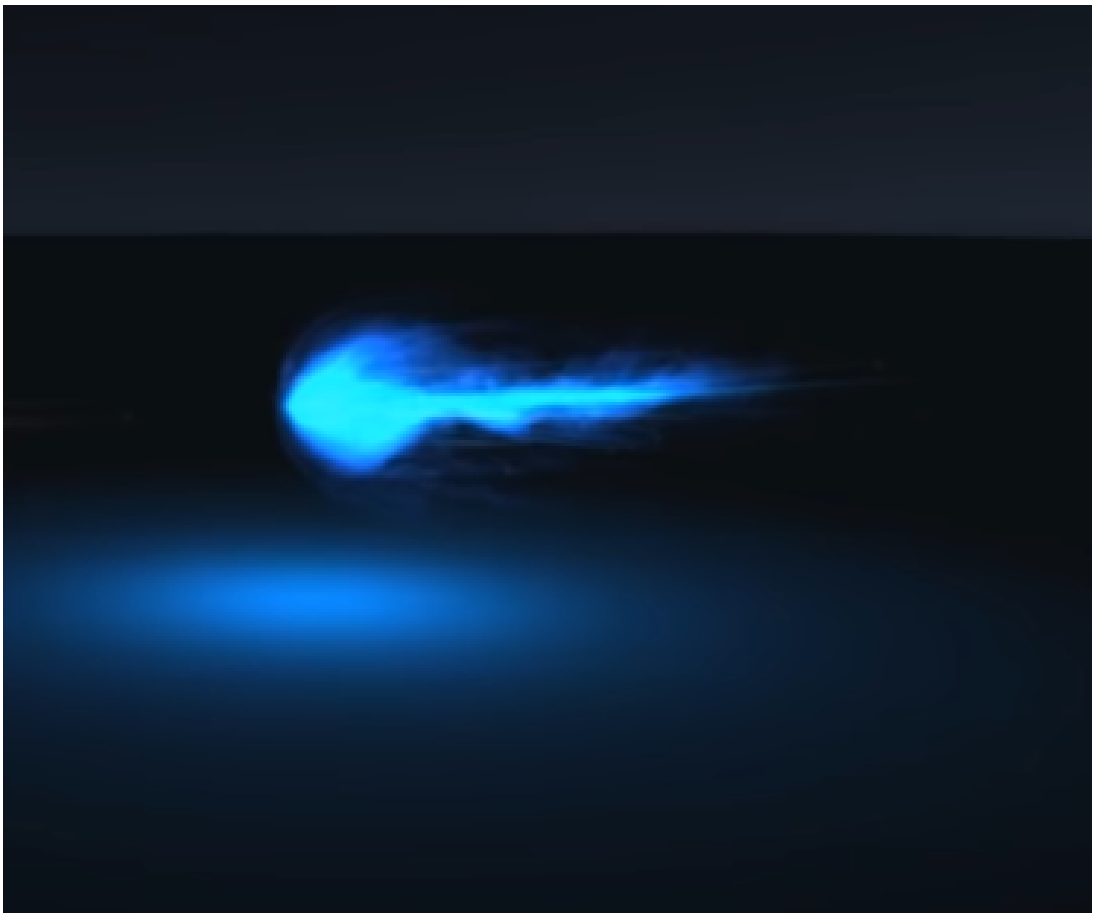


図 2.2: エネルギー波の効果例

本研究におけるエネルギー波はリアルタイム環境での使用を想定しているため、ゲームにおけるエネルギー波の定義方法を参考にしている。 エネルギー波の本体として円柱を定義する。

図 2.1 はゲーム「Fortnite」[9] の中にエネルギー波の例である。図 2.2 は Gabrielaguiarprod[10] が作ったエネルギー波の効果例である。

図 2.3 は、本研究で使用したエネルギー波モデルを示している。

これは blender で作成され、既存の製作技術との互換性を高めるために実験環境にインポートした。この研究の提案では、エネルギー波について次のような前提を置いている。

エネルギー波が x-z 平面に平行な速度ベクトル $\mathbf{p}$ を持つ剛体円柱であると仮定する。エネルギー波が水場に影響を与える可能性のある仮想場を持つと定義した。仮想場は水面と一方的に衝突する。$\mathbf{p}$ は、上部仮想場の円面の中心 C と円柱の中心との差を規定する。

本研究では、計算効率を向上するため計算モデルを単純化する手法を試みた。具体的には，中

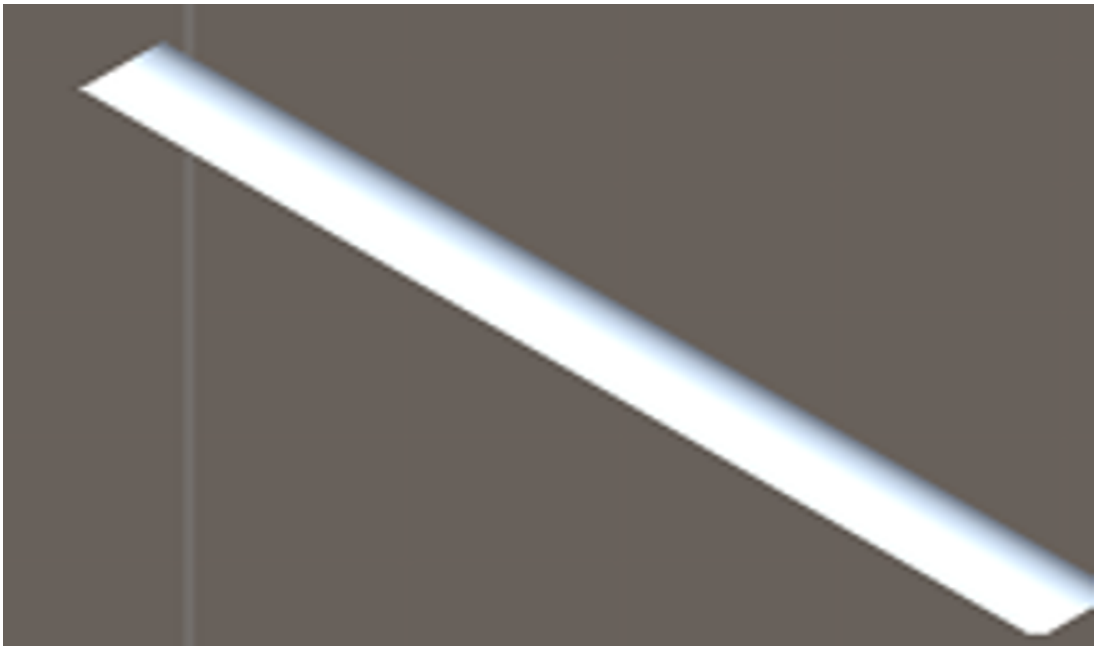


図 2.3: 本研究使っているエネルギー波モデル

心が C で半径 R を持つ球のみが水面に影響と水面を決定する。

アニメーションの効果によって、エネルギー波を囲むフィールドが発生し、物体 (本研究では水面) が仮想半径 R 以内の範囲に入れば、水面頂点の高さを更新する計算を始める。

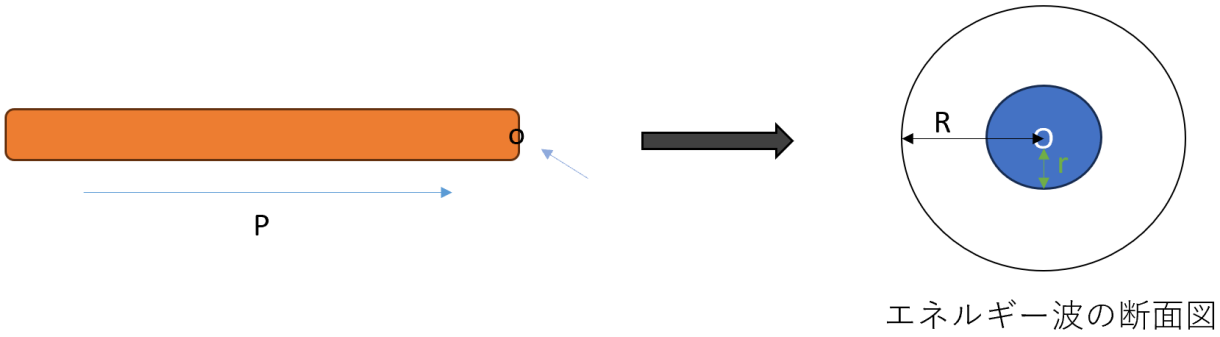



図 2.4: エネルギー波の例

図 2.4 は、本研究で使用したエネルギー波モデルを示している。

- **p** はエネルギー波の進行方向.
- o は計算に関与する頂点または波セクションの円の中心.

- r はエネルギー波断面の円の半径
- R はエネルギー波のフィールドで最も遠い影響位置と円の中心の距離

計算プロセスを簡略化するため、エネルギー波の範囲 R-r で減衰を考慮しない。

## 2.2 水面

### 2.2.1 流体シミュレーション手法の選択

このセクションでは、一般的に使用されている流体シミュレーション手法について説明する。流体シミュレーションには通常、ラグランジュ法とオイラー法の 2 種類がある。

G. Miller ら [11] は最初粒子ベースのアプローチを提案した。現在、水体のシミュレーションによく使われるラグランジュ法には、SPH 法、FLIP 法、PBD 法の 3 つがある。Gingold らは Smoothed Particle Hydrodynamics(SPH)[12] を提案し、Matthias Müller らは [13] は SPH による液体シミュレーションを提案した。Markus Becker らは [14] 弱圧縮性形式流体シミュレーションを提案し Brackbill と Ruppel[15] は、PIC 法における数値散逸問題を解決することを目的とした Fluid-Implicit-Particle(FLIP) 法を導入した。Sulsky ら [16] は材料点法を導入することにより、圧縮性 FLIP 法を弾塑性有限要素法に拡張した。Muller ら [17] は位置更新に基づく粒子運動のシミュレーション手法を提案した。その後、流体シミュレーションの手法として Position Based Fluid[18] を提案した。Hu が提案した Taichi コンパイラ [19][20] を使ってラグランジュ法を検証したところ、精密な結果が得られるが、大規模水面に対して、実行速度が遅いという難点があった。

そこで、この実験では水面にラグランジュ方法を使用しないが、より良い視覚効果を得るために、水の波のエッジに加えることを検討する。

オイラー法では通常、まず頂点を構成する平面を構築する。このようにして、点からなるグリッドを構築する。この格子において、各格子の速度、高さを更新することにより、水面の波紋をシミュレートする。Foster ら [21] はまず、Computational Fluid Dynamics 分野に使っているオイラー法を導入して水面をシミュレートすることを提案した。Stam, J[22] はこれに続き，オイラー法による水面のモデリング方法を改良し，より効率的なモデルを提案した．Foster, N ら

[23] は、より現実的な結果を得るために、オイラー法に基づくセミラグランジュの使用を提案した。O'Brien,J ら [24] がオイラー水面における飛沫流体のモデリング法を提案した。Irving, G ら [25] が大規模な水面をモデル化する際の結合のための方法を提案した。節 1.1 で述べるの SWE[3] 法に加え、Sethian, J[26] は、オイラー水面のシミュレーションにレベルセットを使用することを提案した。Houston, B ら [27] はレベルセットアルゴリズムを改良するために Run-Length Encoded (H-RLE) を提案した。レベルセット法 (Level Set) は水面のシミュレーションをよく使う。

Navier-Stokes 方程式における研究は進展しているが、現在のコンピュータやゲーム機の性能では、上記のような流体シミュレーション手法をリアルタイム環境で適用することが困難である。このため、リアリズムをある程度犠牲にして、よりシンプルな数学モデルを採用し、実行速度が速い結果を得ることが、本研究の目指す方向性である。

### 2.2.2 本研究の水面シミュレーション

本研究では、効率から考えると、SWE モデルに基づいた 3D グリッドを定義して水面と波紋をシミュレートし、直接積分を使って時間ステップを計算する。

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\mathbf{v}) = 0 \tag{2.1}$$

t は時間、h は水の深さ、v は水の速度である。

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{v} + g\nabla h = 0 \tag{2.2}$$

v は流速、g は重力加速度、h は水深である。

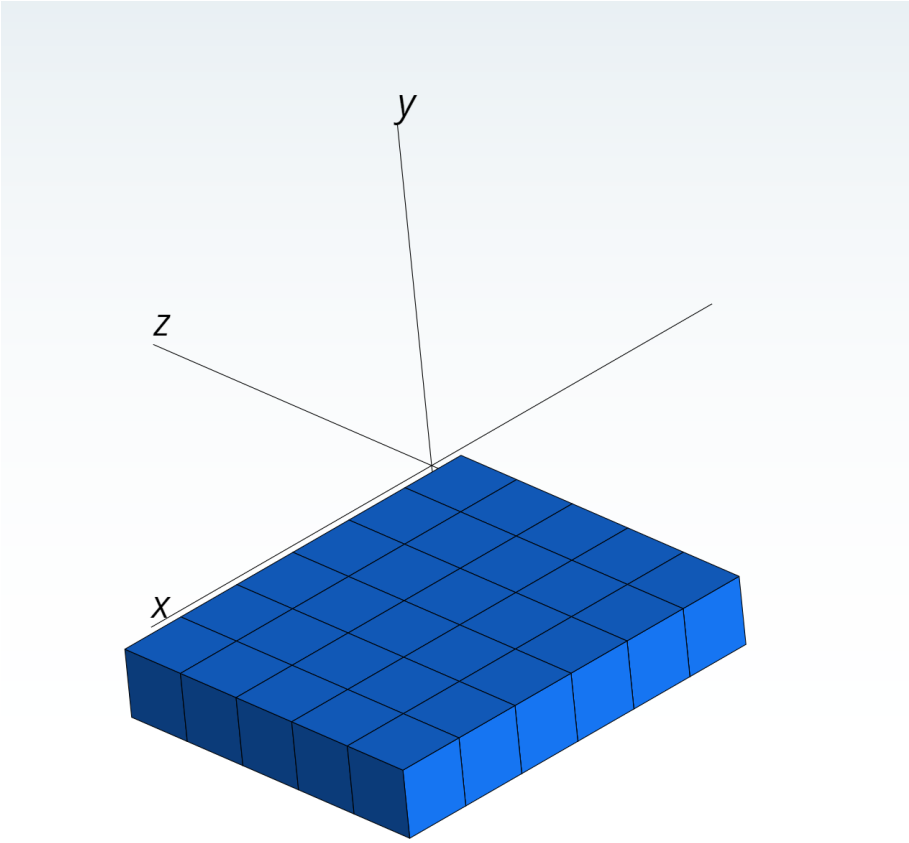



図 2.5: 3D グリッドの構成

図 2.6: 水面の構成

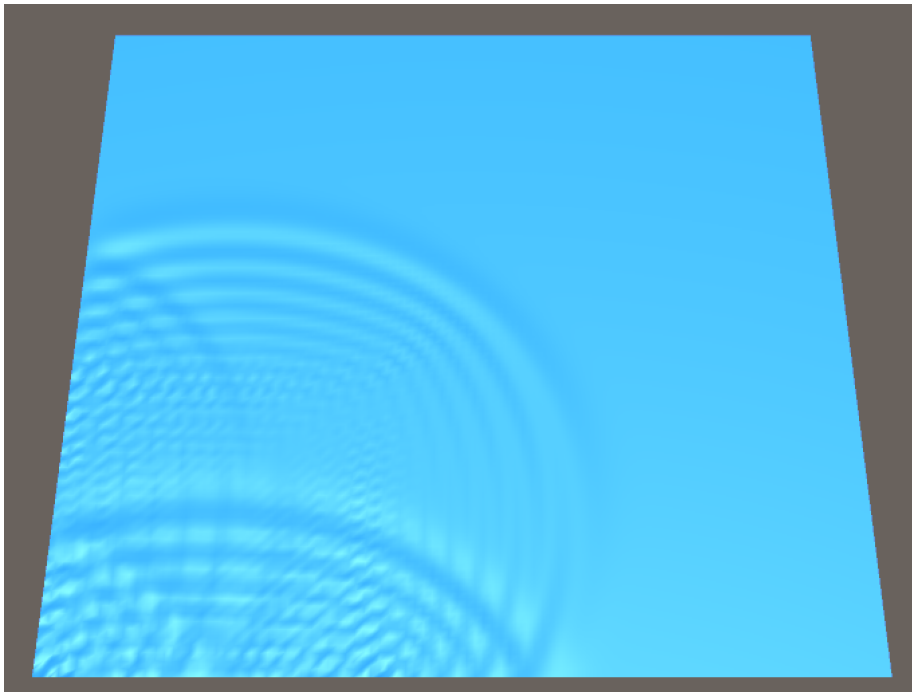


図 2.7: 水面の波紋

図 2.5 は 3D グリッドの構成である。図 2.6 は水面の構成方式である。図 2.7 は SWE での波紋である。

xz 平面に平行な $100 \times 100$ のメッシュを水面と想定し、メッシュ状の頂点の y 成分を高さと設定する。高さを更新した場合や、水が現在のボリュームから外れたりした場合に、現在のフレームの水面の高さを計算するために共役勾配法を使用する。

### 2.2.3 水面波紋の生成とエネルギー波のパワー

SWE[3] は以下の式である。

$$h_i(t_0 + \Delta t) = h_i(t_0) + \beta(h_i(t_0) - h_i(t_0 - \Delta t)) + \alpha(h_{i+1}(t_0) + h_{i-1}(t_0) - 2h_i(t_0)) \quad (2.3)$$

- $h_i$ は水面の任意の頂点 i について、水面の高さ
- $t_0$ は経過時間
- $\Delta t$ は時変量
- $\alpha$ は波の減衰率を表し、$\beta$ は波の強さを示す。$\alpha$ と $\beta$ では通常、SWE の制約を制限し、積分の表示から生じる爆発問題を防ぐために定数を設定する。

本式は、二次元の場合において、次の瞬間 $t_0 + \Delta t$ の高さ $h_i$ を計算するために、$h_i$ とその隣接する格子 $h_{i-1}$、$h_{i+1}$ との関係を示す。

本研究では、水面を通過するエネルギー波に対して異なる $\alpha$ 定数と $\beta$ 定数を割り当てることによって、エネルギー波の強度を調整できる。上限と下限を人為的に設定することで、境界爆発の問題を回避する。

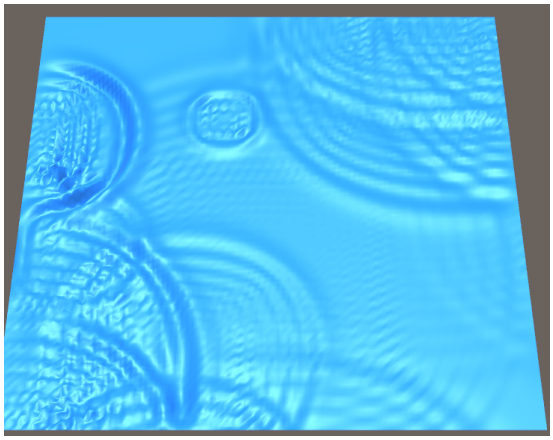


図 2.8: 波の減衰率 $\alpha$=0.995, 波の強さ $\beta$=0.005

図 2.9: 波の減衰率 $\alpha$=0.5, 波の強さ $\beta$=0.005

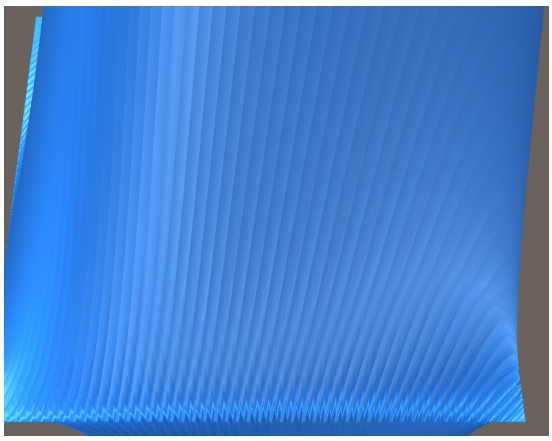


図 2.10: 波の減衰率 $\alpha$=0.995, 波の強さ $\beta$=0.5

図 2.8 は波の減衰率 $\alpha$=0.995, 波の強さ $\beta$=0.005 の場合、波紋の強さを示す。図 2.9 は波の減衰率 $\alpha$=0.5, 波の強さ $\beta$=0.005 の場合、波紋の強さを示す。図 2.10 は波の減衰率 $\alpha$=0.995, 波の強さ $\beta$=0.5 の場合、波紋の強さを示す。モデルの制約を超えたので、爆発が発生する。

## 2.3 エネルギー波プローブ

2.2 節で定義した 2 次元の長方形状の水面は、テクスチャイメージを投影することもできる。そこで、水面上に仮想的な指向性光源 L を定義する。そして、エネルギー波の情報を記録するための仮想プローブを定義する。プローブは半径 R、中心 C で、常に同じ速度ベクトル $\mathbf{p}$ でエネルギー波と一緒に動く。光源 L がプローブを照らし始めると、水面にシャドウマップを生成する。

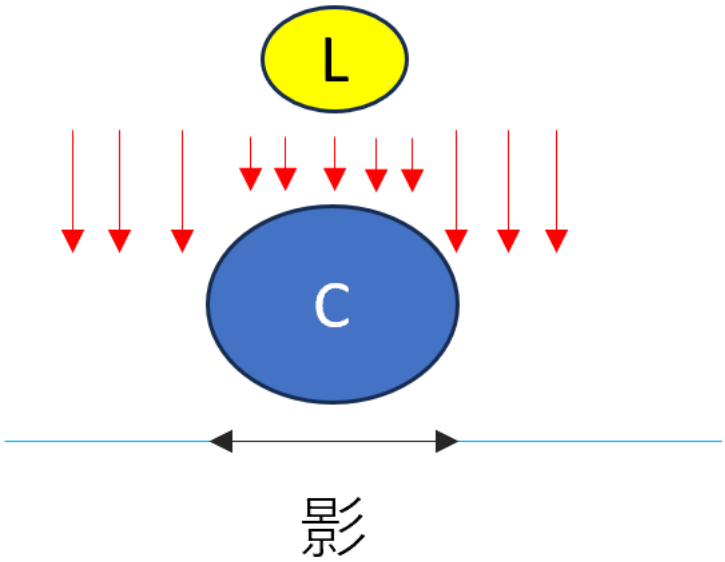



図 2.11: 光源 L がプローブを照らす様子

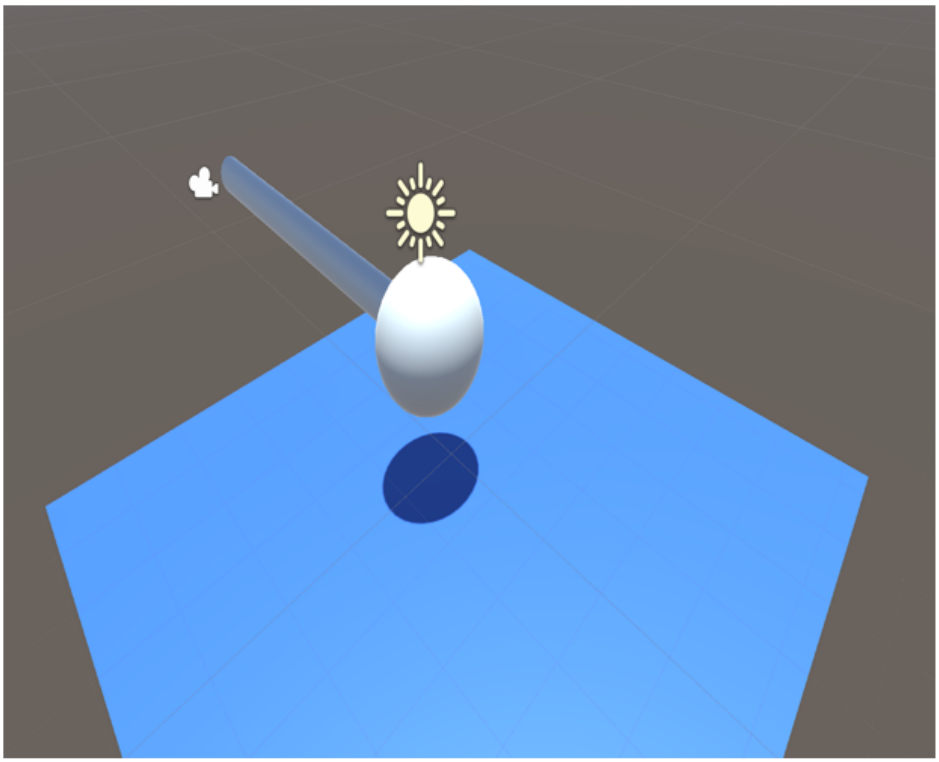


図 2.12: 光源 L がプローブを照らした状況を Unity で描写した様子

図 2.11 で、光源 L はプローブを照らし、水面に影を生成することを示す。図 2.12 は、Unity で運行する時プローブの形状を表示せずに、プローブの影だけレンダリングすることを示す。

光源 L とプローブは独立した層を持っているので、L はエネルギー波に影響を与えない。実際運行する時、プローブの形状とエネルギーの影を表示せずに、画面をレンダリングする。

## 2.4 衝突

Liang ら [28] は 線分クロッピングスペースの使用を提案した. Roth,S[29] は 光をシミュレートするためのレイキャスティングを提案した現在では衝突の検出やゲームでの弾丸発射のシミュレートにもよく使われている。

本研究では、エネルギー波の動きを予測することで不要な計算頂点を取り除き、エネルギー波が水面と衝突するタイミングを特定するために、線分クロッピングスペースとレイキャスティングの技術を採用する。これにより、エネルギー波と水面が衝突する瞬間を効率的に計算する。

Timothy J ら [30] はレイトレーシング法を提案した。この方法は照明効果を得るために光の軌跡を計算することで、光をシミュレートする。Morgan[31] ら光線の計算にプローブ法を用いることを提案した。空間内のプローブによる事前計算を用いて光のマップを得られる。この方法は Global Illumation(GI) 照明にも用いられる。

本研究では、レイトレーシングにおいて光のサンプリングを簡略化するためにモンテカルロ積分を使用するというアプローチを参考にしている。同じような方法が、エネルギー波が水面を通過する際の流体の排除量を簡略化するのにも利用する。

また、本研究では、空間内の点の光情報を記録するためにプロファイルを使用するという考え方を取り入れている。物理シミュレーションで必要な情報を記録するために、エネルギー波の頂点をプローブとして使用し、これによって研究の計算フローを簡略する。

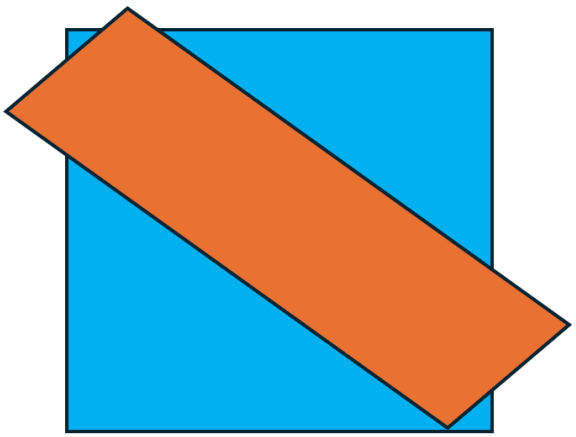


図 2.13: エネルギー波が水面と衝突するときの断面図

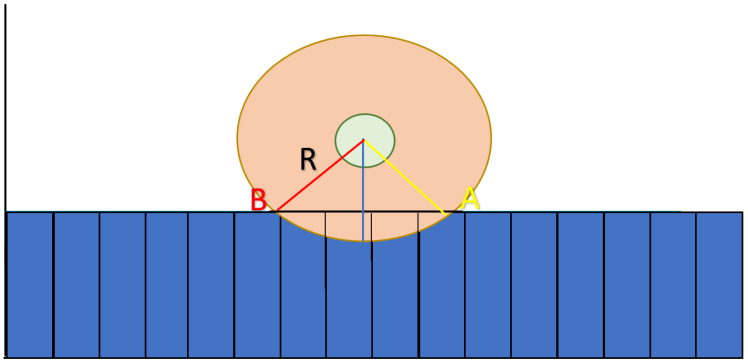



図 2.14: シリングーから水面へのレイを送る

図 2.13 はエネルギー波が水面と衝突したときの断面図である。図 2.14 はシリングーから水面へのレイを送る模式図である。

図 2.14 でのポイント A と B は、水面の高さ計算が必要となる最も遠い位置を示している．第 2.2 節での x-z 平面をバウンディングボックスに変換し、A と B から第 2.1 節で述べた $\mathbf{v}$ の方向ベクトルを持つ光線を送る。

A と B からの光線をそれぞれ「光線 A」「光線 B」と呼称する。方向ベクトル $\mathbf{p}$ と光線 A、光線 B の間の水面の面積獲得する。赤い線が光線 A、黄色い線が光線 B、青い線が第 2.5 節で述べる衝突検査光線を示す。

## 2.5　衝突検査

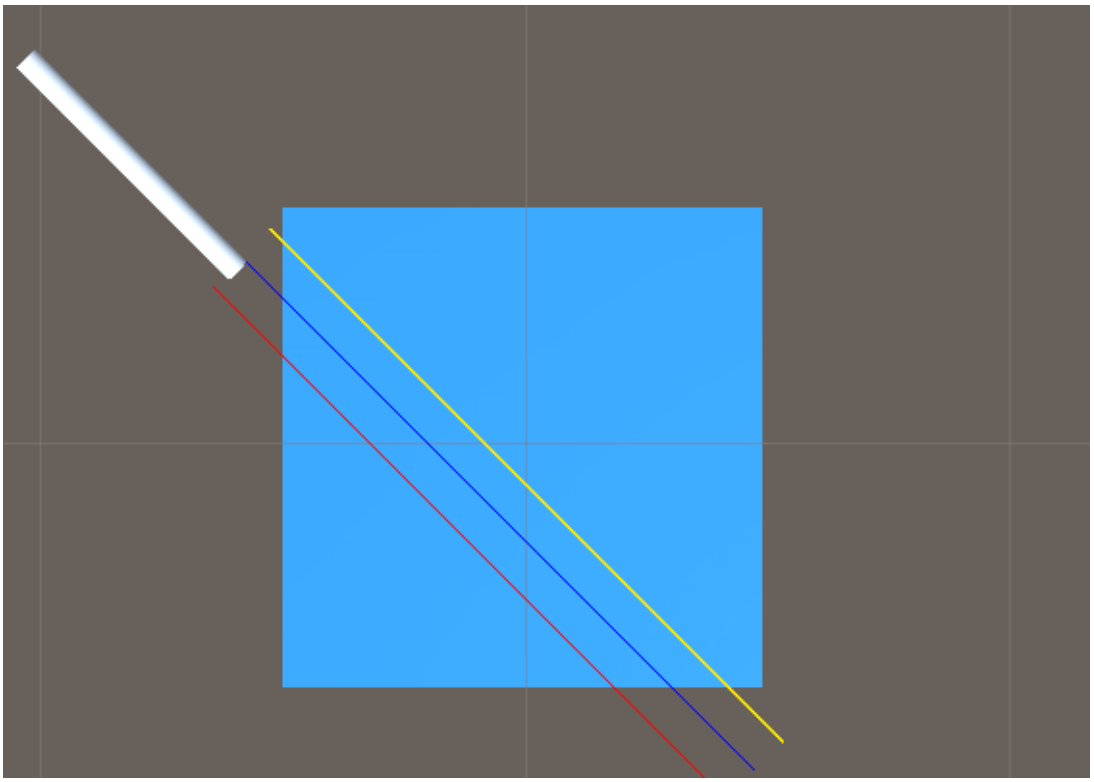


図 2.15: エネルギー波に関する Ray-Casting 光線を送る

図 2.15 はエネルギー波に関する Ray-Casting 光線を示す。

図中の黄線、赤線、青線はそれぞれ節 2.4 の光線 A、光線 B、衝突検査光線である。本研究では、エネルギー波に対する水面からの作用はないものと規定しており、他の力の影響なしに方向が変わらないため、移動を開始する前にそのパスを予測することができる。$O_v$ は点 C を通る xz 平面への投影である。

始点を $O_v$、方向を $\mathbf{p}$ とした直線で衝突検査光線を定義する。各フレームの開始時に、衝突検査光線と水面を形成する正方形のグリッドとの交点が 2 つ存在するかどうかを検出することで、エネルギー波が水面と衝突しているかを判断する。交点が 1 つだけの場合に、エネルギー波と水面の衝突計算を開始する。本研究では、エネルギー波が水面と常に 1 つの交点を持つ状況は対象外としている。

## 2.6　テクスチャーでのカリング

Williams, L[32] はシャドウマップの概念を初めて導入した Dimitrov, R.[33] はケースケイドマップ理論が提案された。この方法は現在、ゲーム分野でもリアルタイム影エフェクトの生成に広く使われている。

本研究では、計算する必要がある頂点の量を減らため、テクスチャー転換を使う。第 2.3 節の仮想光源によって、水面に映る球の影を獲得する。スクリーンショットを撮り、その画像をテクスチャに転換する。第 2.4 でカリングされた領域の頂点の色を比較する。水面のマテリアルの色と違う頂点構成された面積はエネルギー波が水を通過する際に高さを更新する必要がある領域である。

## 2.7　波紋の生成

本研究にとって、Axis-aligned Bounding Box（AABB）[34] などの他の衝突検出方法と比較すると、計算流れを簡単化することにより、本研究の計算フローの一部は SDF モデルに適合する。

物理シミュレーションの分野では、カップリングとは通常、2 つ以上の異なる物理プロセスやシステムを相互に接続し、シミュレーションを通じてそれらの相互作用を考慮できるようにすることを指す。O'Brien ら [24] は流体と剛体カップリング法を提案した。Chen ら [35] は動く物体によって引き起こされる波紋をシミュレーション手法を提案した。本研究では、計算を簡略化するために、エネルギー波が水面に及ぼす影響のみを考慮し、その時点で本研究の計算流れは One-way Coupling を構成する。

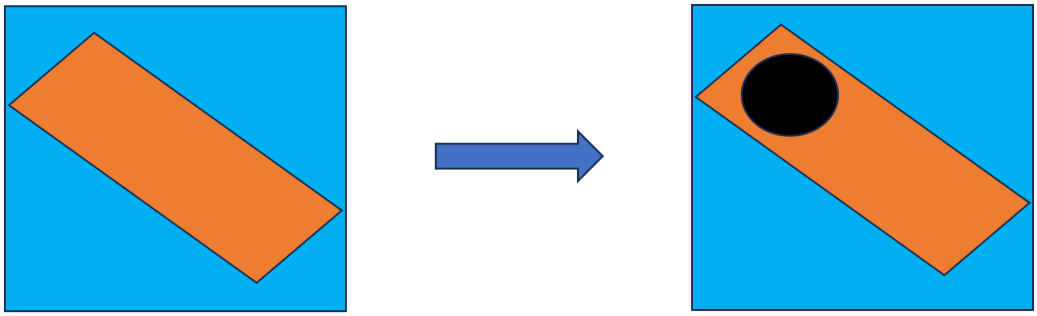


図 2.16: 水面のカリング

図 2.16 に示すのは、カリング処理の結果である。黒色部分は最終計算する必要がある面積である。符号付距離関数を使用して、第 2.7 節での面積の中の頂点 $\mathbf{P}_{ij}$ は、エネルギー波の点 C 分別計算する。$H$ は水面の頂点と円の中心 C の距離となる。

$$\phi(\mathbf{P}_{ij}) = \sqrt{||\mathbf{P}_{ij} - \mathbf{C}||^2} - R. \tag{2.4}$$

$$\mathbf{H}_{ij} = \phi(\mathbf{P}_{ij}). \tag{2.5}$$

$\mathbf{P}_{ij}$ は黒色部分の頂点の集合であり、$\mathbf{C}$ は使われているエネルギー波のプローブの中心である。$R$ はプローブの半径である。

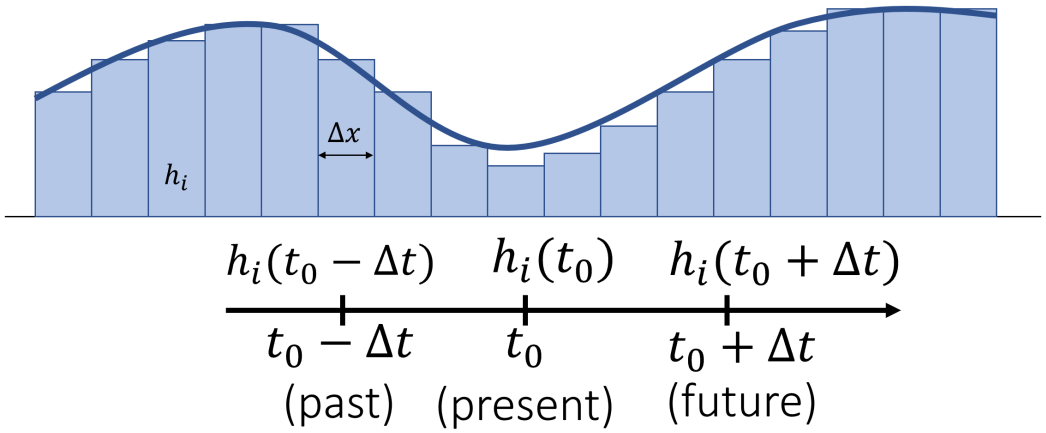



図 2.17: SWE の高さの計算

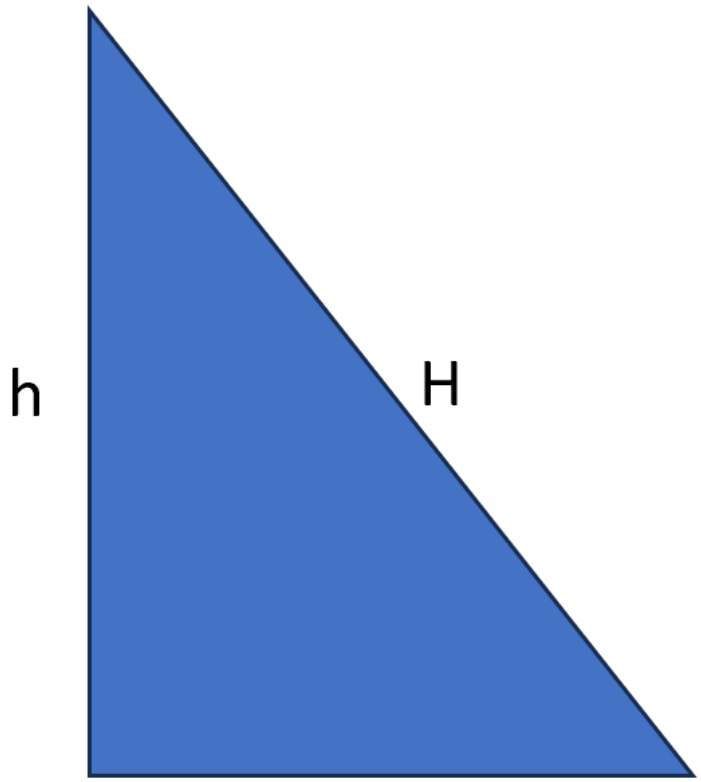



図 2.18: 2.4 の H と SWE の h の関係

図 2.17 は、Wang の Intro to Physics-Based Animation Lesson 10 Surface Waves Page 10[36] の SWE 方程式で水面の頂点の高さ $h$ の更新処理を示したものである。図 2.18 で、水面の頂点と円の中心 C の距離 $H$ と円の中心 C から水面までの距離 $h$ の関係を示す。

$H$ によって、SWE の高さを更新する。その結果、H は SWE の高さ h と同じではないので、プロセスを簡略化するために、両者を同じものとして扱うことにした。

# 第 3 章

# 検証

## 3.1　環境

### 3.1.1　ハードウェアとソフトウェアの環境

本実験では、以下のハードウェア環境を使用した。

- Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz
- SanDisk Extreme Pro NVME
- NVIDIA GeForce RTX 2080 Ti

本実験は、以下のシステムで実行した。

- Windows 10 Pro Windows Feature Experience Pack 1000.19052.1000.0
- unity 2022.1.6f1

本実験は C#言語を使用し、Unity ゲームエンジンのスクリプトレイヤー上で実行した。

### 3.1.2　実験環境

xz 平面に、モデルの中心の座標を (0,0,0) として平面モデルを置き、水面を表す青いテクスチャを使う。モデル内の 100 × 100 頂点を以下の方法でバインドする。

円柱モデルを位置 (-8, 1, 6) に置き、回転 (0,45,90)、スケール (0.5, 3.0, 0.5) 移動速度を 1.0f、移動方向を (0,45,90) とした。

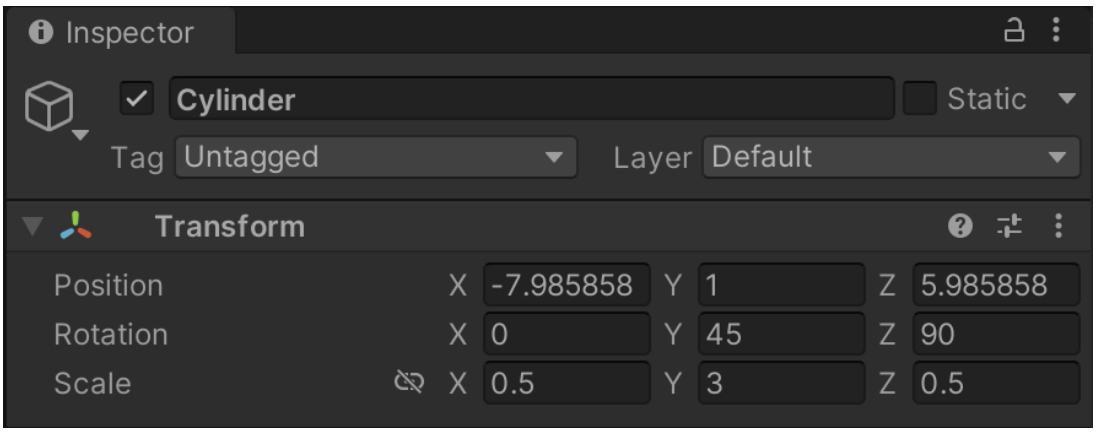



図 3.1: エネルギー波の設置

図 3.1 はエネルギー波の基本数値設定を示している。

SWE 方程式の時間ステップは 8 とし、1 フレームあたり 8 回実行した。節 2.2.3 で述べた波の減衰率 $\alpha$ と波の強さ $\beta$ は、それぞれ 0.995 と 0.005 と設定した。

## 3.2 パフォーマンス実験

従来の物理シミュレーション法を用いて、すべての水面の頂点に対して計算を行う。本研究では、行動コースを予測することで、CPU を使用した頂点のカリング手法を提案する。この CPU による頂点カリングの手法に基づき、GPU でシェーディングされたピクセルの色を比較することにより、さらに頂点をカリングする方法を採用している。

これら 3 つのアプローチを性能比較する。以下は、一般手法、Ray-Casting 手法、Pixel-Culling 手法と呼ばれる。

- 一般手法は節 2.2 の水面の全て頂点を計算する手法。
- Ray-Casting 手法は節 2.4 の述べていた方式を通じて、光線でエネルギーの移動経路を予測し、水面をカリング手法。
- Pixel-Culling 手法では節 2.6 と節 2.7 の述べていた方式を通じて、影により、水面をカリング手法。

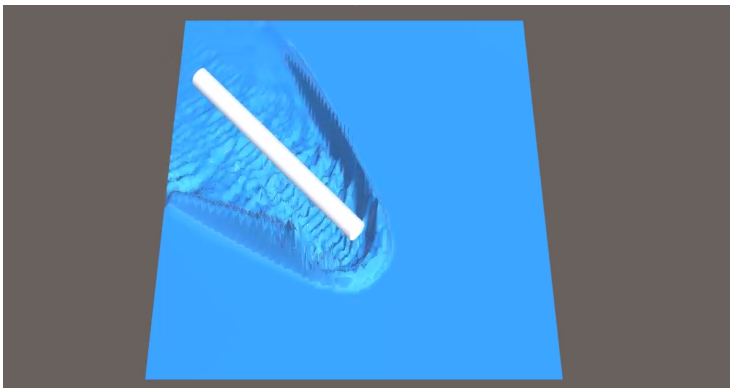


図 3.2: 一般手法の実装画面

図 3.2、図 3.3、図 3.4 は一般方式、Ray-Casting 方式、Pixel-Culling 方式それぞれの実行画面を示している。

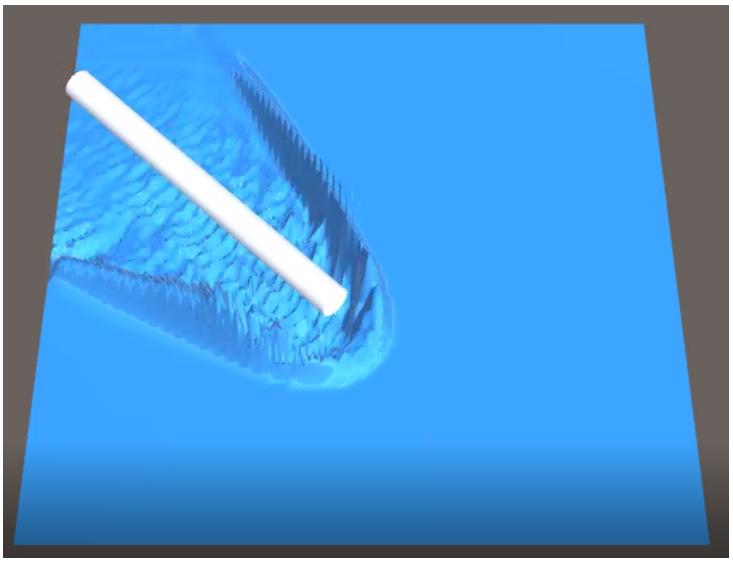


図 3.3: Ray-Casting 手法の実装画面

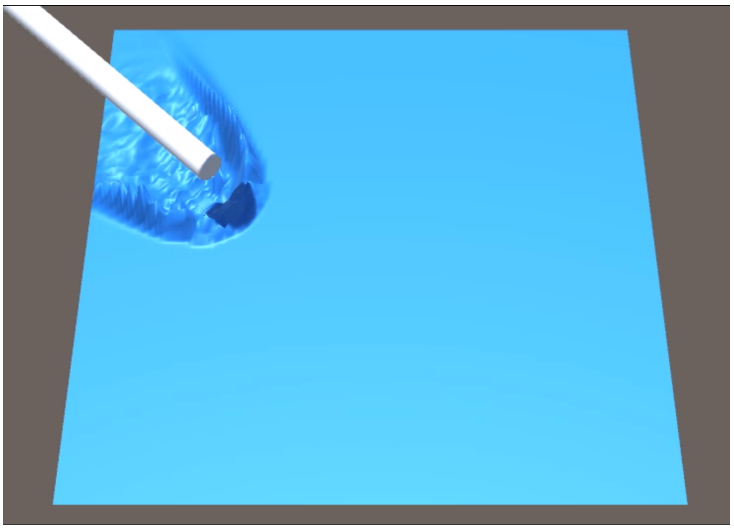


図 3.4: Pixel-Culling 手法の実装画面

表 3.1: 3 つ方式の効率一覧

| 方法 | FRAME 数 | 計算頂点数/FRAME | 平均 FPS |
|---|---|---|---|
| 一般 | 113 | 80000 | 18.0 |
| Ray-Casting | 109 | 49504 | 19.5 |
| Pixel-Culling | 49 | 37224 | 11.6 |

標準的な方法では、対照群を設け、設定に従い $8 \times 100 \times 100$ 頂点の 113 フレームを使用した。

平均 FPS は 18 だった。FPS が 30 を下回る理由は複数あり、スクリプトレイヤーで物理シミュレーションを実行することで、物理エンジンレベルでの関数呼び出しが増え、CPU のオーバーヘッドが増加する。C#言語の自動メモリ管理は便利だが、C++ などの言語での手動メモリ管理に比べてシステムのオーバーヘッドが増加する。デバッグメッセージの関数 Debug.Log() の呼び出しはシステムリソースを消費する。

したがって、この実験では平均 FPS を絶対値ではなく相対値として参照している。アルゴリズムの有効性と効率性を判断するため、主に計算される頂点数を基準にしている。Ray-Casting 手法では CPU を用いた場合、109 フレームを実行し、計算された頂点数は 49,504 で、平均 FPS は 19.5 である。これは、標準的な方法と比較して計算頂点数を大幅に削減し、FPS の効率を相対的に向上させ、この方法の有効性を証明している。

Pixel-Culling 手法では、GPU を使用して 1 フレームあたり 37,224 の固定頂点で 49 フレームを実行し、平均 FPS は 11.6 であった。計算頂点数は前の 2 つの方法と比較して最小である。アニメーションの結果にも、シミュレーションで示されたような顕著な異常は見られない。この方法は冗長な平面を除去するために使用され、その実現可能性を示した。

## 3.3 ロバスト性

本実験は、異なる角度から水面にエネルギー波が入射した際に、提案された計算手法がシステムの安定性を保持できるかを検証する目的で行った。陽解法積分と大きなステップサイズを採用しているため、陽解法積分における爆発的な挙動を避けるために定数を用いて制約を設けた。このアプローチにより、他のシナリオでもシステムが安定するかどうかを検証し、本研究の一般性を試すことが可能である。

図 3.5 は、節 3.2 中に Pixel-Culling 手法でエネルギー波が水面を通過した場合、水面は初期状態に戻る例を示す。

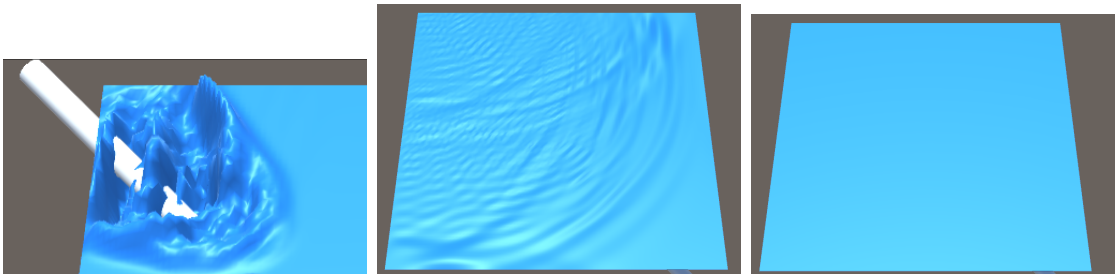
実験を通じ、対照群として、節 3.1.2 の条件下で、エネルギー波が水面を通過した後、水面システムを安定させることができる。

図 3.6 は、エネルギー波の回転角度を (0,45,80) になる場合、水面の変化となる。

実験を通じ、エネルギー波は水面に平行から水面に入れるまで移動方向変えた後、水面が最初

(a) (b) (c)

図 3.5: Pixel-Culling 手法で水面は初期状態に戻る例

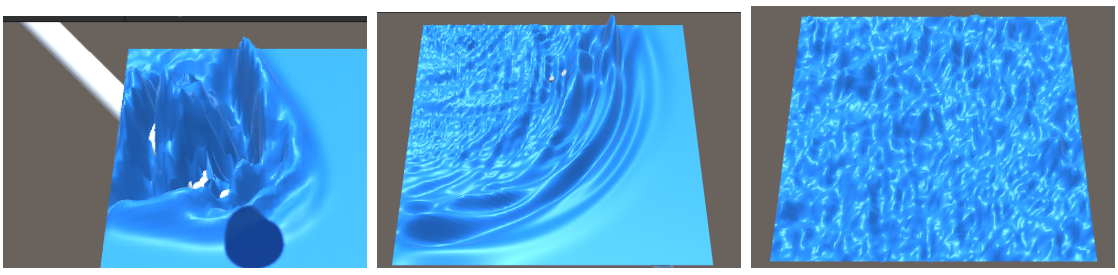


(a) (b) (c)

図 3.6: Pixel-Culling 手法でエネルギー波の角度が変えた後水面状態の例 1

状態に戻れる。本研究では、波の減衰率 $\alpha$ をコントロールすることで、エネルギー波の強度を定義する。アニメーションの効果によれば、エネルギー波が水面に近ければ近いほど、発生する波紋は大きくなる。

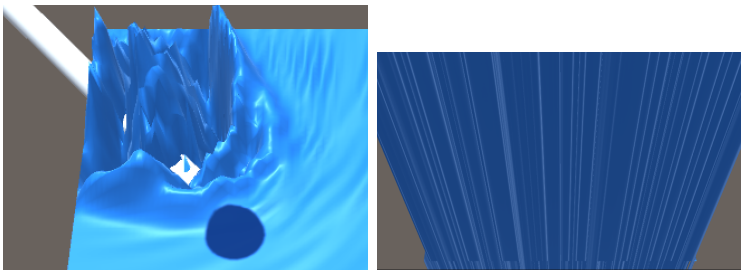
本研究の計算手順でこの効果が得られるかどうかを確認するため、波の減衰率 $\alpha$ を 1.000 と 1.005 に設定し、角度を変えながら実験を行った。



(a) (b) (c)

図 3.7: Pixel-Culling 手法でエネルギー波の角度が変えた後水面状態の例 2

図 3.7 は、エネルギー波の回転角度を (0,45,80) になる場合、波の減衰率 $\alpha$ を 1.000 に設定すると水面の変化となる。



(a) (b)

図 3.8: Pixel-Culling 手法でエネルギー波の角度が変えた後水面状態の例 3

図 3.8 は、エネルギー波の回転角度を (0,45,80) になる場合、波の減衰率 $\alpha$ を 1.005 設定すると水面の変化となる。

実験を通じ、エネルギー波の角度が (0, 45, 80) のとき、システムは波の減衰率 $\alpha = 1.000$ という条件で安定に戻ったが、動的なままであった。波の減衰率 $\alpha$=1.005 のとき、システムは安定に戻ることができなかった。a が大きくなるにつれて、より顕著な波紋が生じる。

## 3.4 考察

節 3.2 に示した実験結果より、本提案では、エネルギー波が水中を通過し波紋を生成する現象を、物理シミュレーションを用いてゲームエンジン内で実現することに成功した。経路予測とピクセルの選択的処理を取り入れることにより、計算に必要な頂点の数を削減している。

節 3.3 に示した実験結果より、提案するシミュレーション処理は、ある程度の安定性を持ち、小さな調整も可能である。

Pixel-Culling 手法は頂点の計算量を大幅に削減したが、速度が低下するという欠点がある。これは実装方法に起因する問題である。まず、C#言語を使用しているため、C++ の手動メモリ管理に比べて C#の自動メモリ管理がメモリ消費を増加する。さらに、本研究は Scipt Layer で実装されており、グラフィック API を直接使用するよりも関数呼び出しが多くなり、関数呼び出しの増加はメモリ消費の増加につながる。最後に、全ての処理を GPU 上で行うのではなく、ゲー

ムエンジンの関数を CPU 側で呼び出し、その後 GPU 側で処理を行うため、CPU と GPU 間の通信が頻繁に発生し、計算負荷が増大する。

理論的には、Pixel-Culling 手法は頂点計算を大幅に減らし、計算中フレームバッファのレンダリングパイプラインデータを再利用できるため、GPU 側で直接 C++ とシェーダ言語を使用して実装すれば、本研究の実装方式より高速度を実現できる。

## 3.5 新規性と有用性

本研究では、計算される頂点の数を大幅に削減し、レンダリングパイプラインで既に計算されたデータを再活用することにより、現実には存在しないがゲーム内で頻繁に見られる現象を、シンプルな物理モデルを用いてシミュレートすることを目指す。このアプローチの利点は、シミュレーションの計算効率が理論上向上することにある。一方で、計算を大幅に簡素化するため、リアリズムと安定性がある程度犠牲になるというデメリットがある。本研究の新たな取り組みは、2つの点がある。一つ目は、通常ゲーム内でアニメーションとして表現される現象を、物理シミュレーションの手法で再現しようとする試す。二つ目は、レンダリングパイプラインと組み合わせて、レンダリングデータの再活用により計算量を削減しようとするアプローチである。ゲームにおけるリアルタイムレンダリングの標準的な方法は、オフラインレンダリングの方程式を大幅に単純化し、その効率を模倣することで成り立っている。本研究の有用性は、ゲーム内で物理シミュレーションを実装するための別の手法を試すことを提案する。

## 3.6 パフォーマンス改善と再現性との間にトレードオフ

パフォーマンスを向上させるために、より低レベルでの実装を採用することは、理論上可能なアプローチである。しかし、できるだけ少ない関数呼び出しに依存し、メモリ管理を手動で行うことは、現在の実装手法より、実装の複雑性を大幅に増加する。ゲームエンジンが大規模化し、役割が明確に分かれている現在、全ての部分を自作することは難しい。そのための解決方式として、Enlighten[37] や Yebis[38] など、市場にすでに存在するソリューションを参考し、本研究の実装をユーザーにシンプルな GUI や API インターフェイスを提供するミドルウェアとしてカプセル

化することは、パフォーマンスを向上させつつ実装の複雑さを軽減する参考になる方向性である。

# 第 4 章

# まとめと展望

本研究は、アニメーションやゲームで頻繁に見られる現象、すなわちエネルギー波が水面を通過し、水面がエネルギー波の影響を受けて波紋を生成する様子を、物理シミュレーションを通じて再現することを目的としている。検証のため、基本的な計算プロセス、Ray-Casting を用いた水面の抽出、Pixel-Culling を通じた計算プロセスのさらなる最適化がそれぞれ実施した。結果として、提案されたアプローチが、必要な頂点数を減少させつつ、受け入れられるシミュレーション結果を実現することを確認した。ただし、基礎となる API の実験は行われておらず、理想的な効率は達成されない。また、陽解法積分と大きなステップサイズを用いた SWE モデルを大幅に簡素化して使用しているため、計算プロセス全体の安定性には改善の余地を残っている。将来的には、DCC ツールを活用して、さらに多くのエネルギーモデルをこのフレームワークに適用する予定である。ラグランジュ法を導入し水滴をシミュレートすることも、水滴の視覚効果を向上させるための一つの手段である。さらに、エネルギー波が水面に入射するようなシーンを増やすこと、仮想光やプローブを利用したランダムサンプリングによる影の効果で波紋の形状を変化させることも目指している。現在は、GPU の演算結果を CPU で処理するのが難しい課題ですが、GPU を用いた物理シミュレーションの研究が活発になっており、将来的には GPU を活用した水面の物理シミュレーションがこの研究の進展方向の一つである。

# 謝辞

本研究を進めるために多くのご指導やアドバイスをしてくださった先生方や研究室のメンバーに感謝を述べたいと思います。

研究生から大学院への 3 年間、指導してくださった渡辺先生に心から感謝の意を表したいと思います。渡辺先生は私が興味を持つ研究分野を自由に試行し、探求できる環境を提供してくださいました。また、学会ごとに貴重なアドバイスをいただき、論文を発表する前には、その修正方法についても多くの助言をいただきました。

大学院に進学してから研究科に移るまでの 3 年間、大変お世話になった阿部さんには深く感謝しています。どの学会でも、私の発表のポイントを的確に抽出し、日本語が流暢でない私に対しても建設的なアドバイスをしてくださいました。論文発表の前の修正作業にも協力していただき、多くの重大なミスを未然に防ぐことができました。

本論文の主査を務めていただいた柿本先生、副査の菊池先生、榊原先生には、この調査研究で大変お世話になりました。質問や提案をするたびに、彼らは常に私の研究に新たな展開をもたらしてくださいました。

研究室の同期たちには、ミーティングでの議論や日常生活での出会い、助け合いに感謝します。この 3 年間はとても充実していました。

大学院時代、授業やプロセスについて丁寧に指導してくださった奥野さんに感謝の気持ちを表します。

レンダリングに関する授業と演習を担当してくださった Yan 先生には、グラフィックスの世界に触れるきっかけをいただき、心から感謝しています。グラフィックス入門とリアルタイムレンダリングの授業を通じて、コンピュータグラフィックスの基礎を学ぶことができました。また、オフラインでのパストレーシングレンダラーの完成や、OpenGL を通じたレンダラーフレームワークの実装など、様々な学びを得ることができました。

Wang 教授の物理シミュレーションのコースを受講し、グラフィックス分野での物理シミュレーションに触れることができたことに感謝しています。このコースは非常に厳しく、失敗や理解の難しさを乗り越えながら、少しずつ前進しました。数式の理解やプログラミングにはしばしば 1 週間を要しましたが、試行錯誤の末、物理シミュレーションを少し理解できるようになりました。この学びは、私の研究や将来の職業に対する方向性と基礎を形成してくれました。

ゲームエンジンのコースで指導してくださった Wang さんにも感謝します。このコースを通じて、最新のゲームエンジンについて直感的に理解することができました。また、学術的なモデルから産業界での実用化への移行における妥協点やプロセスの理解にも役立ちました。毎回 2 時間の授業は、ゲームを楽しむこと以上の喜びを私に与えてくれました。

私の生活と研究を支えてくれた mabu、piku、shi にも感謝します。

# 参考文献

[1] Matt Pharr and Randima Fernando. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation (Gpu Gems)*. Addison-Wesley Professional, 2005.

[2] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, Vol. 378, pp. 686–707, 2019.

[3] KASS M. and MILLER G. Rapid, stable fluid dynamics for computer graphics. *In Proc. of ACM SIGGRAPH'90*, pp. 49–57, 1990.

[4] H. Wang, G. Miller, and G. Turk. Solving general shallow wave equations on surfaces. *In Proc. of the ACM Siggraph/Eurographics Symposium on Computer Animation*, pp. 229–238, 2007.

[5] Anis Zarrad. Game engine solutions. In Dragan Cvetković, editor, *Simulation and Gaming*, chapter 5. IntechOpen, Rijeka, 2018.

[6] 阿部雅樹. ボリュームレンダリングを用いたエネルギー波のリアルタイム形状変形. 学部卒業論文, 東京工科大学メディア学部ゲームサイエンスプロジェクト, 2008.

[7] 阿部雅樹. エネルギー波表現のリアルタイムレンダリング. 修士論文, 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, 2010.

[8] 阿部雅樹, 渡辺大地. エネルギー波表現のリアルタイムレンダリング. 芸術科学会論文誌, Vol. 9, No. 3, pp. 93–101, 2010.

[9] Fortnite Dragonball. Fortnite Dragonbal. `https://www.fortnite.com/news/goku-powers-up-fortnite-x-dragon-ball-your-power-is-unleashed`. 参照: 2024.2.05.

[10] GabrielAguiarProd. Visual EffectsforGames. `https://www.gabrielaguiarprod.com/`. 参照: 2024.2.05.

[11] G. Miller and A. Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, Vol. 13, No. 3, pp. 305–309, 1989.

[12] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, Vol. 181, pp. 375–398, 1977.

[13] Matthias Müller, David Charypar, and Markus Gross. Particle-based fluid simulation for interactive applications. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 154–159, Aire-la-Ville, Switzerland, 2003. Eurographics Association.

[14] Markus Becker and Matthias Teschner. Weakly compressible sph for free surface flows. pp. 1–8, 2007.

[15] J. U. Brackbill and H. M. Ruppel. Flip: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, Vol. 65, pp. 314–343, 1986.

[16] D. Sulsky, S.-J. Zhou, and H. L. Schreyer. Application of particle-in-cell method to solid mechanics. *Computer Physics Communications*, Vol. 87, pp. 236–252, 1995.

[17] M. Muller, B. Heidelberger, M. Hennix, and J. Ratcliff. *Journal of Visual Communication and Image Representation*, Vol. 18, No. 2, pp. 109–118, April 2007.

[18] Miles Macklin and Matthias Muller. Position based fluids. *ACM Transactions on Graphics (TOG)*, Vol. 32, No. 4, 2013.

[19] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, Vol. 38, No. 6, p. 201, 2019.

[20] Yuanming Hu, Jiafeng Liu, Xuanda Yang, Mingkuan Xu, Ye Kuang, Weiwei Xu, Qiang Dai, William T. Freeman, and Frédo Durand. Quantaichi: A compiler for quantized simulations. *ACM Transactions on Graphics (TOG)*, Vol. 40, No. 4, 2021.

[21] N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, Vol. 58, No. 5, 1996.

[22] J. Stam. Stable fluids. In *Proc. of ACM SIGGRAPH '99*, pp. 121–128, 1999.

[23] N. Foster and R. Fedkiw. Practical animation of liquids. In *Proc. of SIGGRAPH '01*, pp. 23–30, 2001.

[24] J. F. O'Brien and J. K. Hodgins. Dynamic simulation of splashing fluids. In *Computer Animation '95*, pp. 198–205, 1995.

[25] G. Irving, E. Guendelman, F. Losasso, and R. Fedkiw. Efficient simulation of large bodies of water by coupling two and three dimensional techniques. In *Proc. of ACM SIGGRAPH '06*, p. 38, 2006.

[26] J. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[27] B. Houston, M. B. Nielsen, C. Batty, O. Nilsson, and K. Museth. Hierarchical rle level set: A compact and versatile deformable surface representation. *ACM Transactions on Graphics*, Vol. 25, No. 1, pp. 151–175, 2006.

[28] Y.D. Liang and B.A. Barsky. A new concept and method for line clipping. *ACM Transaction on Graphics*, Vol. 3, No.1, pp. 1–22, 1984.

[29] S. Roth. Ray casting for modeling solids. *Computer Graphics and Image Processing*, Vol. 18, pp. 109–144, 1982.

[30] Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, Vol. 21, No. 3, pp. 703–712, 2002.

[31] Morgan McGuire, Mike Mara, Derek Nowrouzezahrai, and David Luebke. Real-time global illumination using precomputed light field probes. *Association for Computing Machinery*, 2017.

[32] Lance Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, Vol. 12, No. 3, p. 270 – 274, aug 1978.

[33] R. Dimitrov. *Cascaded Shadow Maps*. NVIDIA Corp, 2007. Developer Documentation.

[34] Panpan Cai, Chandrasekaran Indhumathi, Yiyu Cai, and Jianmin Zheng. *Collision Detection Using Axis Aligned Bounding Boxes*, pp. 1–14. Springer, 2014.

[35] J. X. Chen, N. da Vitoria Lobo, C. E. Hughes, and J. M. Moshell. Real-time fluid simulation in a dynamic virtual environment. *IEEE Computer Graphics and Applications*, Vol. 17, No. 3, pp. 52–61, 1997.

[36] Wang Huamin. GAMES 103 Lesson10SurfaceWaves. `https://slides.games-cn.org/course/GAMES103-10-wave.pptx`. 参照: 2024.02.05.

[37] Silicon Studio. Enlighten. `https://www.siliconstudio.co.jp/middleware/enlighten/`. 参照: 2024.2.19.

[38] Silicon Studio. Yebis. `https://www.siliconstudio.co.jp/middleware/yebis/`. 参照: 2024.2.19.

# 発表実績

## ポスター発表

- 蘇路文, 阿部雅樹, 渡辺大地 エネルギー波と水の物理ソルバーの研究, 映像表現・芸術科学フォーラム, 2023
- Luwen Su, Masaki Abe and Taichi Watanabe, A study of energy wave with water physical solver, NICOGRAPH International 2023, 2023

## 口頭発表

- 蘇路文, 阿部雅樹, 渡辺大地, GPU を用いた水面とエネルギー波の相互作用に関する研究, DiGRA JAPAN 2023 Summer, 2023