卒業論文

ホイヘンスの原理の視覚化プログラムの開発

関西学院大学理工学部

情報科学科 西谷研究室

3518 村上 大貴

2017 年 3 月

概 要

情報や通信に関する技術の総称を ICT(Information Communication Technology) という. 近年,ICT を教育に活用する動きが活発であり, このような流れを受けて, 政府は 2020 年を目標に全ての中学, 高等学校で,1 人 1 台のタブレット端末を導入した ICT 授業を実現するという目標を立てている.

タブレット端末の利点として, インタラクティブな教材を使用できる点が挙げられるが, このような教材が効力を発揮する教科の一つに物理がある. 物理現象は, 文章や数式による記述では非常に理解しづらいものが存在する. 例えば, 高校物理で学習する回折, 反射, 屈折といった波の性質は上記のような物理現象に相当すると考える. これを直観的に理解させ, 学習を促進する手段として, 物理現象の視覚化が適していると考えた. 本研究では, 反射, 屈折, 回折といった波の性質の理解を助けるプログラムの作成を目的とした.

回折, 反射, 屈折の性質を視覚化するためには, 多数の波を生成し, それらに重ね合わせの原理を適用しなければならない. そこで複数の点源から波を生成した上で, 重ね合わせの原理によって生じる干渉を視覚化するプログラムを作成した. また, 平面波の描写処理が非常に重かったため, 描画領域のピクセル数を擬似的に変更することにより, これを大幅に軽減した.

研究の結果,Processing 言語を用いて, 波の干渉と回折現象を視覚化し, シミュレーションするプログラムを作成した. これにより複数の波が干渉した際の複雑な変位変化や, 波長によって回折の度合いが変わる様子を視覚的に理解できるようになった.

今後の課題は, 反射角と屈折角の描写が理論通りに描写できなかったため, これを描写できる新たな手法を考えなければならない. また完成したプログラムも, 教育現場で使用できる程のデザイン性を有していないため, さらなる改良が必要である.

目 次

第1章　序論

情報や通信に関する技術の総称を ICT(Information Communication Technology) というが, 近年 ICT を教育に活用する動きが活発である. 例えば,ICT 活用授業による学力向上に関する調査では,ICT を活用した授業をおこなった教員の 97.3%が児童生徒の学力向上に効果があると認めているデータが得られており, このような動きはますます加速すると思われる [1]. このような流れを受けて, 政府は 2020 年を目標に全ての中学, 高等学校で,1 人 1 台のタブレット端末を導入した ICT 授業を実現するという目標を立てている [2].

タブレット端末を学習に活用することが出来れば, 従来指導が難しいとされていた教科も分かり易く指導を行える可能性が生まれる. そのような状況でタブレット端末の特性を活かした学習コンテンツの作成が喫緊の課題であると考える.

タブレット端末の利点としてインタラクティブな教材を使用できる点が挙げられるが, このような教材が効力を発揮する教科の一つに物理がある. 物理現象は, 文章や数式による記述では非常に理解しづらいものが存在する. 例えば, 高校物理で学習する干渉, 回折, 反射, 屈折といった波の性質は上記のような物理現象に相当すると考える. これを直観的に理解させ, 学習を促進する手段として, 物理現象の視覚化が適していると考える.

本研究では, 干渉, 反射, 屈折, 回折といった波の性質の理解を助けるプログラムの作成を目的とする.

本書の構成は以下の通りである．まず次章では, どのような波の振る舞いを理解するためのソフトを開発するかを明確にするため, 関連する物理学の知識を簡単に紹介する．次に 3 章では，開発したソフトの設計や仕様を理解してもらうために，簡単に使用法と振る舞いを示す．4 章ではソフトの実装において工夫した点や問題となった課題をどのように解決したかの議論を行っている．5 章では本研究で完成させることができなかった反射, 屈折の法則の視覚化プログラムの実装状況を示し, 処理の問題点について考察している.

第2章　物理学的基礎知識

2.1　波の要素

ある点での振動が他の点へと伝わっていく現象を**波**という. 波を伝える物質を**媒質**といい, 振動によって最初に波が起きた点を**波源**という.

図 2.1 は最も基本的な波である**正弦波**である. 波形の中で最も高い所を**山**, 最も低い所を**谷**という. 隣り合う山同士, 谷同士の間の距離を波長 λ[m] といい, 波源から山の高さ, もしくは谷の深さを波の**振幅**という.

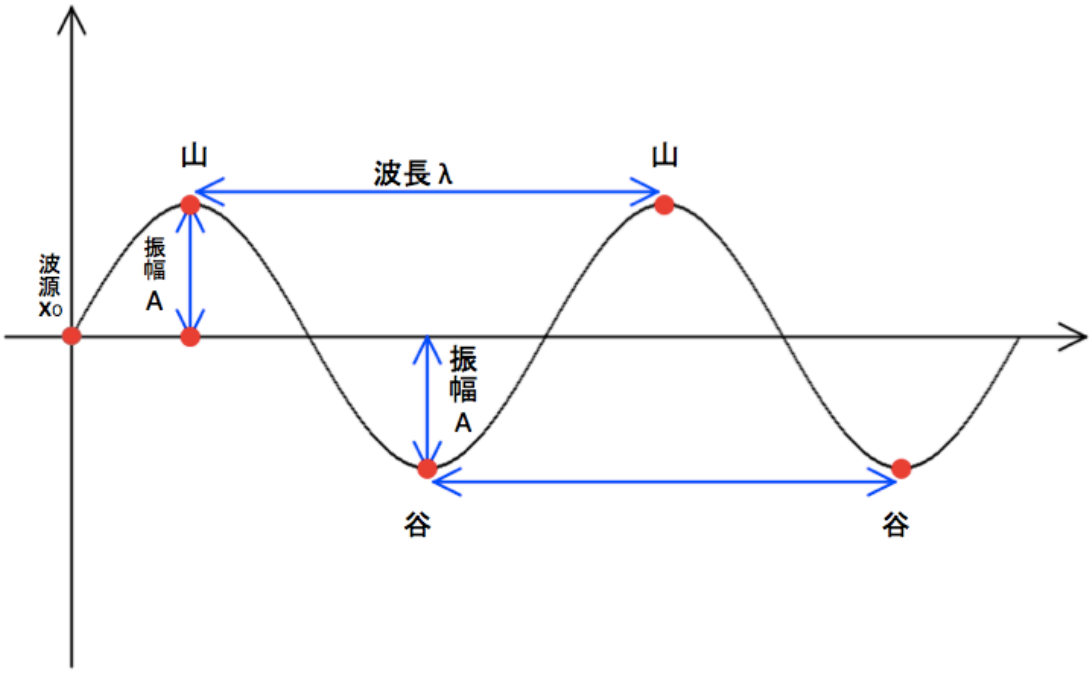


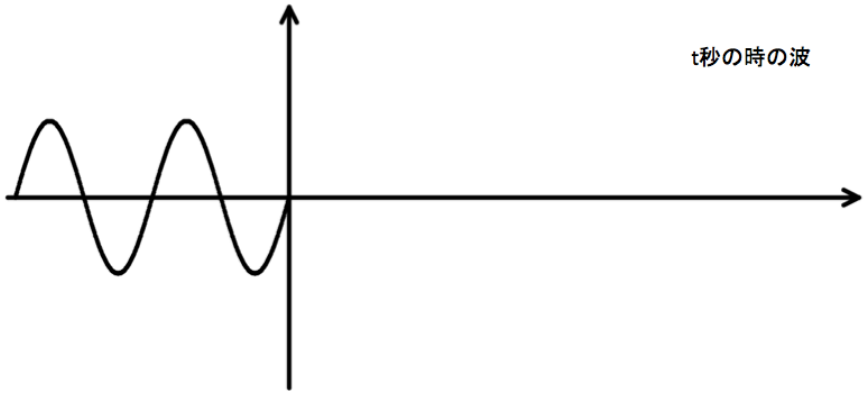
図 2.1: 波の振幅, 波長を示した図.

2.2　波の速度と振動数と周期の関係

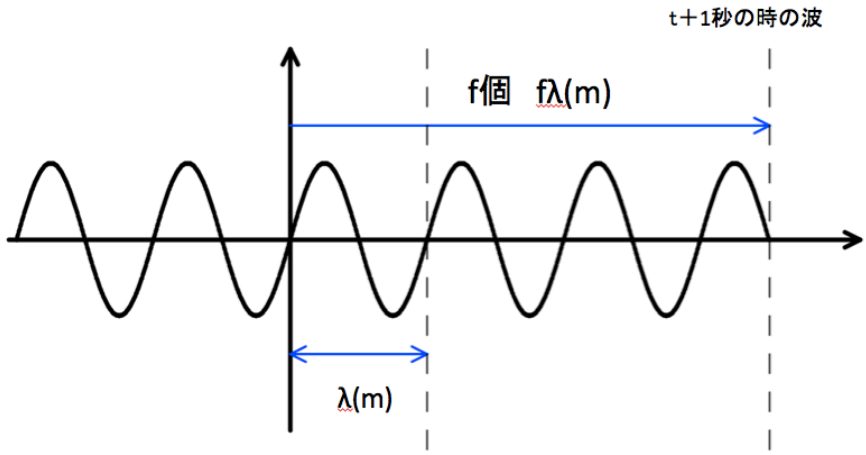
1 波長分の波が 1 秒間に発生する回数を**振動数** f[1/sec] という. 波の速さ v[m] は式 (2.1) と表すことができる.

$$v = f\lambda \tag{2.1}$$

図 2.2 は波の速さ v と振動数 f の関係を示した図である.



(a) t 秒の時の波.



(b) t+1 秒の時の波.

図 2.2: 波の速さ v, 振動数 f の関係を示した図.

また 1 波長分の波が発生するまでに要する時間を**周期** T という. 周期 T と振動数 f には式 (2.2) の関係が成り立つ.

$$T = \frac{1}{f} \tag{2.2}$$

波に対し空気抵抗や摩擦力などの他の外力を一切適用しなければ, 周期ごとの波形は完全に一致する. 今回作成するプログラムは教材として用いることや, 処理の複雑さに伴う描画速度の低下を考慮し, 他の外力を一切適用しないとして波を描写するため, これ以降の記述は全て他の外力が一切適用されていない状態での波を考えているものとする.

2.3 正弦波の変位の計算方法

任意の地点かつ時間における正弦波の変位を計算する際には, 以下に挙げる要素が必要である.

1. 波長
2. 波源からの距離
3. 現在の時間と波源が生成された時間との差

全ての要素を同時に考慮することは難しいので,1-2 の要素と 3 の要素を分けた上で正弦波の変位を計算する方法を説明する. 計算方法の手順を以下に示す.

1. 正弦波の任意の地点における位相変化が波源 (角度は 0°) の位相変化よりどれだけずれているかを, 波源から任意の地点までの距離と波長から算出する.
2. 目標とする時刻と, 波源の位相が 0° に生成された波の位相のずれを算出する.
3. 手順 2 で求めた位相のずれから手順 1 で求めた位相のずれを引いたものを y とし,$\sin(y)$ の値に振幅を掛け合わせる.

手順 1 の詳細を以下に示す. 波源の位相が 0° である時の正弦波で考える. 任意の地点を A とする. 図 2.3 は波源から地点 A までの距離を表したものである. 波源から目標地点 (地

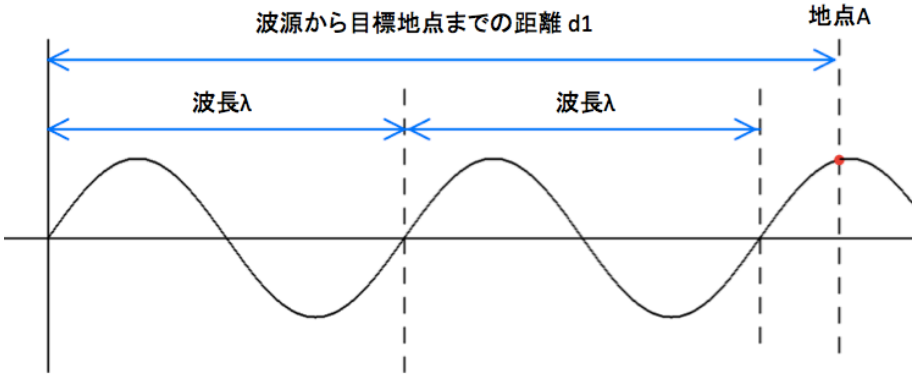


図 2.3: 正弦波における波源と地点 A の関係.

点 A) までの距離を d1 とする. この時, 波は 1 波長ごとに同じ形をしていることから, 地点 A の位相 (0°∼360°) は d1 を λ で割った値に 360 を掛けることで求めることができる.

図 2.4 はある時刻 $t0$ と, 正弦波が $\frac{\lambda}{4}$ 分だけ進んだ時刻 $t1$ の正弦波の状態を表したものである. 時刻 $t0$ の時, 波源の正弦波の変位は $\sin(90°)$ である. これに対し, 時刻 $t1$ では目標地点 A の変位が $\sin(0°)$ となり, 波源の変位は $\sin(90°)$ となる. つまり目標地点 A の位相変化は波源よりも 90° 遅れていることが分かる. よって手順 3 では波源と目標地点の位相のずれを減算しなければならない.

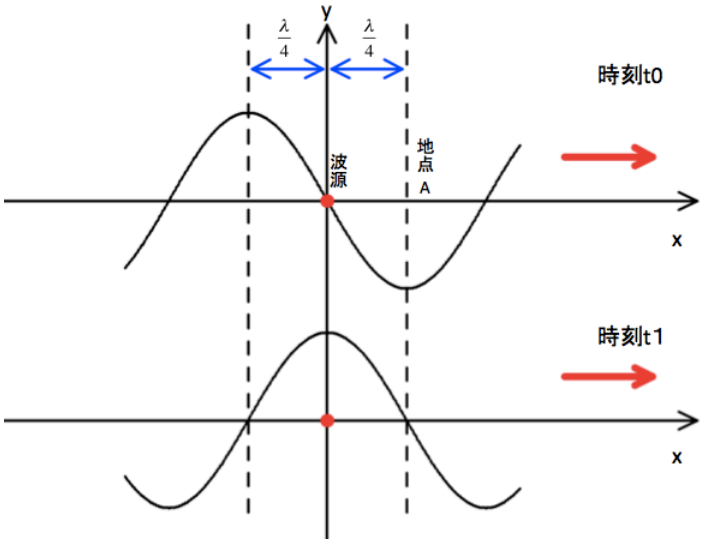


図 2.4: 波源の位相変化と地点 A の位相変化の関係.

手順 2 の詳細を以下に示す. まず正弦波の波源の位相が 0° である時刻 $t0$ から $\frac{1}{4}$ 周期分経過した後の時刻 $t1$ に波源が生成された例を考える. 図 2.5 の上側の青色の波は時刻 $t0$ の正弦波 w0 であり, 下側の青色の波は時刻 $t1$ の時の正弦波 w1 である. また下側の薄い赤色の波は $t0$ の正弦波 w0 である. w1 の波は w0 で考えるためには角度のずれを足さなければならない. w0 の任意の点の位相は w1 で対応する点の位相よりも $\frac{1}{4}$ 周期, つまり 90° だけ先に進んでいることが分かる. つまり w1 を w0 に対応させるために手順 3 で値を 90° 分足さなければならない.

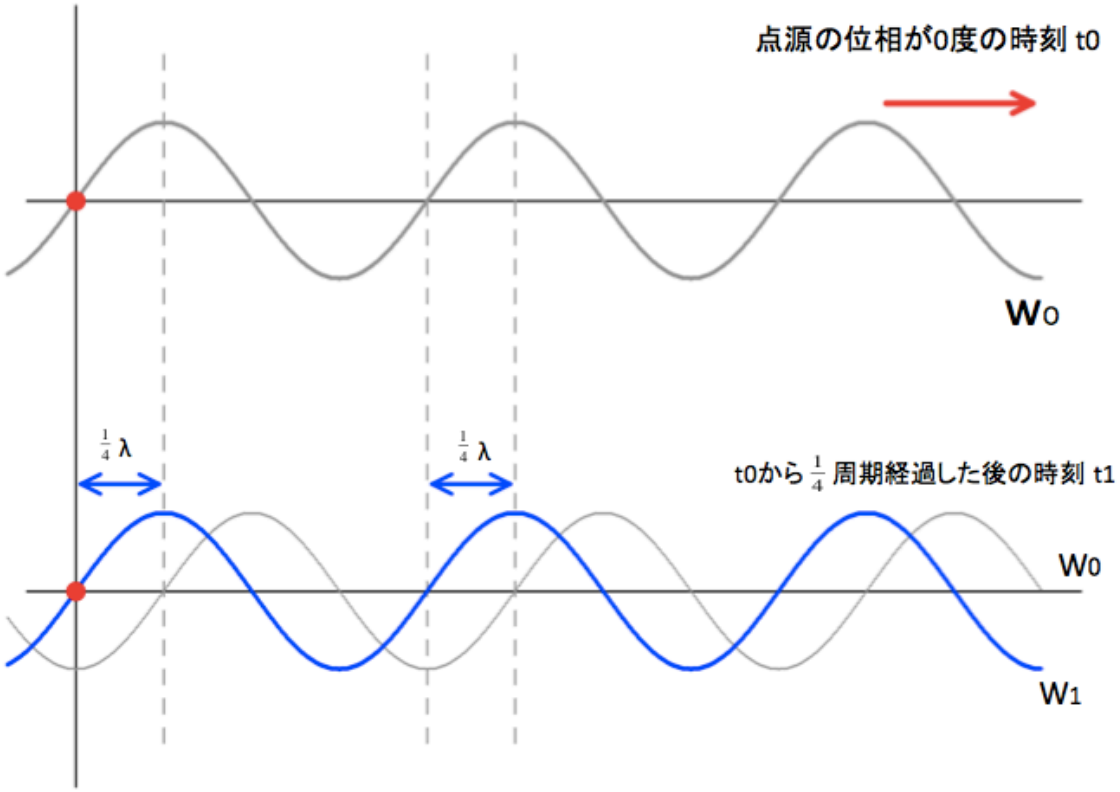


図 2.5: 経過時間による変位の変化を表した図.

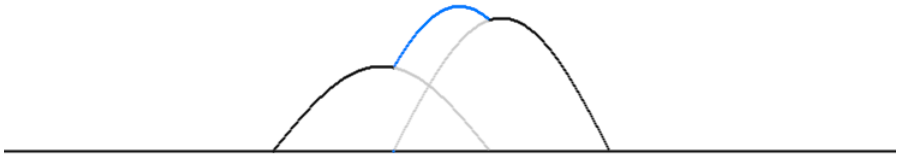
2.4　波の独立性と重ね合わせの原理

図 2.6(a) は別の波源から生じた 2 つの波が互いに近づいている様子である.

図 2.6(b) は 2 つの波が重なっている様子である. 水色の波の変位は,2 つの波の変位を足し合わせたものである. このように複数の波が重なり合ってできた波を**合成波**という. 合成波の変位が, 重なっているそれぞれの波の変位を足し合わせたものになることを, **重ね合わせの原理**という. また, 複数の波が重なり合って弱めあったり強めあったりする現象を波の**干渉**という.

図 2.6(c) は 2 つの波が重なった後, 離れていく様子である.2 つの波は一度重なったにもかかわらず, 図 2.6(a) の時と同じ波形を保っている. このような性質を**波の独立性**という.

(a) 2 つの波が重なる前.



(b) 2 つの波が重なっている時.

(c) 2 つの波が重なった後.

図 2.6: 波の独立性と重ねあわせの原理.

2.5　ホイヘンスの原理

同じ時刻に同じ状態の点を結んだ線または面を**波面**という. ある波面とその隣の波面との間隔は波長 λ である. 波面が平面であるものを**平面波**といい, 球面状のものを**球面波**という.

ホイヘンスの原理とは, 任意の波面上の全ての点がそれらを波源とする球面波（素元波）を発生させ, 素元波の共通に接する面が次の瞬間の波面を形作るとする理論である.

図 2.7, 図 2.8 はホイヘンスの原理により平面波, 球面波が新たな波面を形作る様子を示したものである. 赤い線が波面, 赤い点が素元波を生成する点, 青い線が素元波としている.

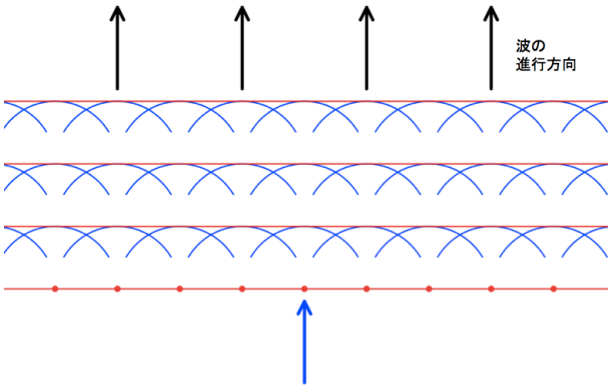


図 2.7: 平面波.

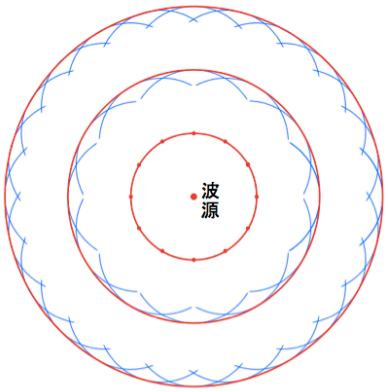


図 2.8: 球面波.

2.6　反射の法則

まず, 入射波のある端点が点 B_1 に到達した時, 点 B_1 から素元波が生成される. この時, 素元波は入射波と同じ媒質中を進むため, 入射波と同じ速度で進行する. 次に, 入射波の異なる端点が遅れて A_2 点に到達する．その時，B_1B_2 点と A_1A_2 点の長さは同じであり, 三角形 ACD と三角形 ABD は合同となる. この事から, 入射角 i と反射角 j が等しくなることがわかる．これを反射の法則という.

$$i = j \tag{2.3}$$

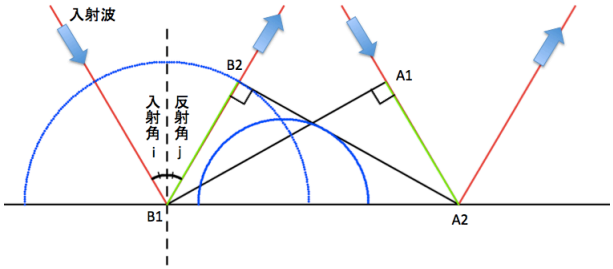


図 2.9: 反射の法則.

2.7　屈折の法則

波の速度は, 媒質の種類によって変化する. 波がある媒質から別の媒質に進んでいくことで波の速度が変化し, 媒質の境界面を境として波の進む方向が変わることを**波の屈折**という.

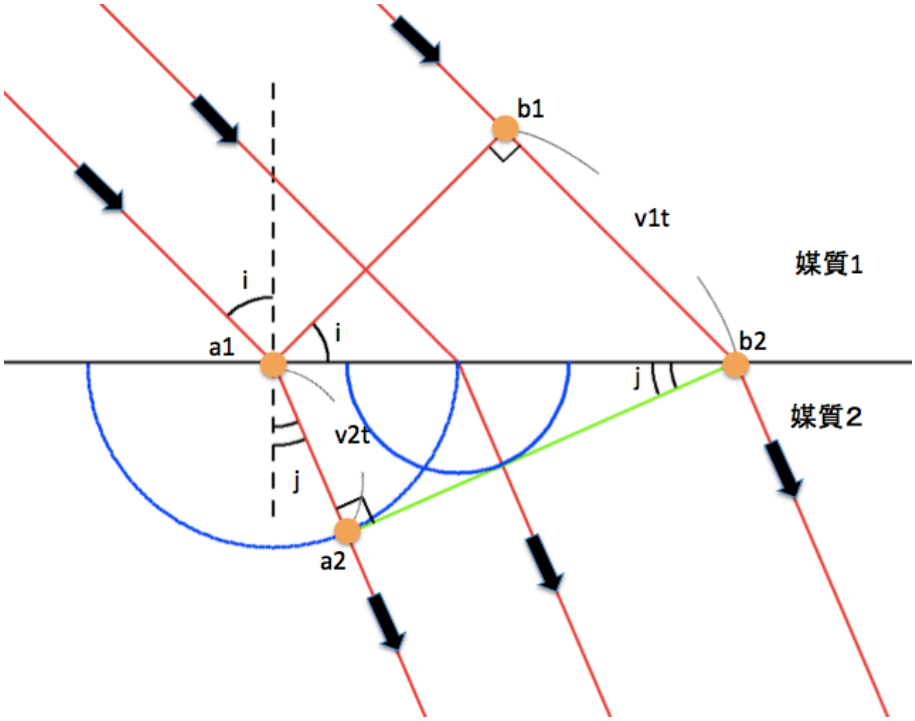


図 2.10: ホイヘンスの原理による屈折の説明.

図 2.10 はホイヘンスの原理の考え方を用いて波が屈折する仕組みを示したものである.

媒質 1,2 における波の速さをそれぞれ v_1, v_2 とし, 波面 a_1b_1 上の点 b_1 が b_2 に達するまでの時間を t とすると,

$$b_1b_2 = v_1t \tag{2.4}$$

が成り立つ. この時, 時間 t に a_1 から出た素元波は v_2 の速度で広がるため,

$$a_1a_2 = v_2t \tag{2.5}$$

となる.

式 (2.4), 式 (2.5) から

$$v_1t = a_1b_2\sin(i) \tag{2.6}$$

$$v_2t = a_1b_2\sin(j)$$

が導出される．式 (2.6) と屈折によって波の振動数は変化しないことから式 (2.7) が成り立つ.

$$\frac{\sin(i)}{\sin(j)} = \frac{v1}{v2} = \frac{\lambda 1}{\lambda 2} \tag{2.7}$$

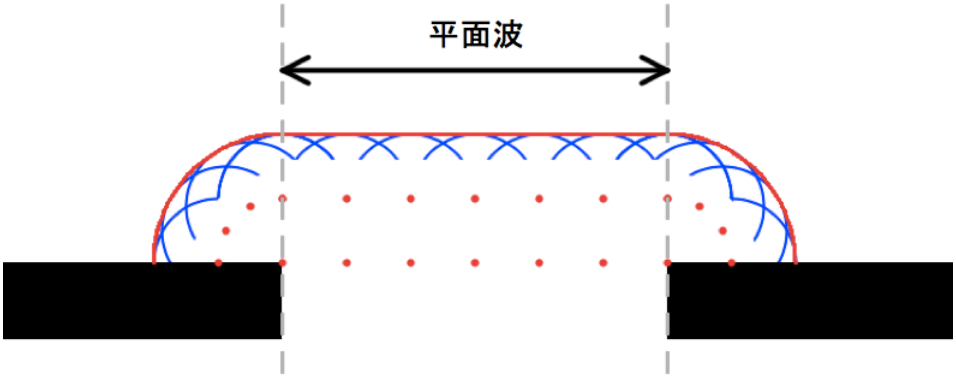
式 (2.7) の $\frac{v1}{v2}$ や $\frac{\lambda 1}{\lambda 2}$ は媒質 1,2 の物質によって決まる一定の値である. これを**屈折率** n と定義すると式 (2.8) が成り立つ. これを**屈折の法則**という.

$$\frac{v1}{v2} = n_{12} \tag{2.8}$$

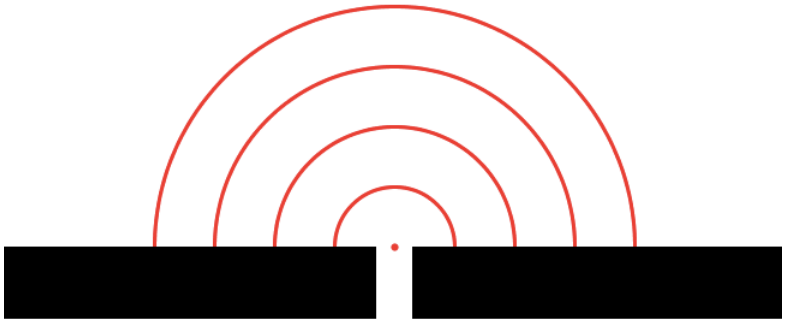
2.8 回折現象

2.8.1 ホイヘンスの原理による回折現象の説明

波が障害物の裏まで回り込む性質を波の**回折**という．図 2.11(a) は障害物間の間隔が広い時の波の様子を示している. この時, 隙間の中央部分は平面波, 両端部分は円形波として伝わる．一方, 図 2.11(b) のように障害物間の間隔が狭い時は, 平面波の領域が小さくなり, 波の形は円形波に近くなる.



(a) 間隔が広い時.



(b) 間隔が狭い時.

図 2.11: 隙間の間隔の広さによる波の形の差異.

2.8.2 実際の回折現象

図 2.11(a),(b) では波同士の干渉や波の波長を考慮していない. 実際は図 2.11(a) のような状況だと, 隙間の両端部分は各波の位相のずれで打ち消しあってしまうため, 回折現象はほとんど見られない. 隙間の間隔が小さく, 波長の長さが長い方が回折現象は起こりやすい. 実際の回折現象をシミュレーションした結果は 3.2 節に記す.

第3章　開発結果

本章では，波の干渉と回折現象をプログラムによって視覚化した実行結果を記述する．本研究では以下に挙げる物理現象の視覚化プログラムに取り組んだ.

1. 重ね合わせの原理を適用した波の干渉.
2. 波の回折現象.
3. 反射の法則.
4. 屈折の法則.

これらのうち,1 と 2 の現象は目標通りに視覚化することに成功したが, 3 と 4 の法則の視覚化については未完成の部分を残す形となった. 3 と 4 の未完成部分の考察については 5 章に記す.

3.1　波の干渉の視覚化

図 3.1 は指定した複数の点源から生成される円形波の位相変位をシミュレーションし, 視覚化を行うプログラムである. このプログラムには draw モードと check モードという 2 つのモードが実装されており,draw モードから check モードに移行するには C キー (check の頭文字).check モードから draw モードに移行するときは D キー (draw の頭文字) を押せばよい.

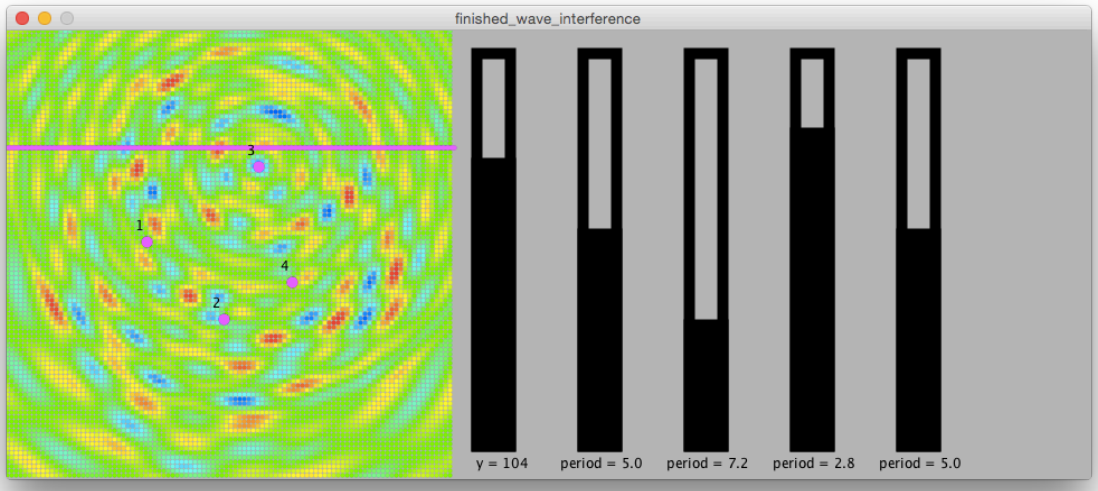


図 3.1: 波の位相変位を視覚化したプログラムの画面.

3.1.1 draw モード

このモードは波源から生じる円形波を描写するモードである. プログラム起動時の画面が図 3.2 である. 画面左側の黒い領域が波を描写する領域, 画面右側には check モードで確認する y 座標の位置を操作できるスライダーが配置されている. 図 3.2 の状態で黒い領域上のいずれかの場所をクリックすると, クリックされた座標に波源が生成される. 波源が生成されたあと, 画面右側にはその波源の周期を変更できるスライダーが生成される.

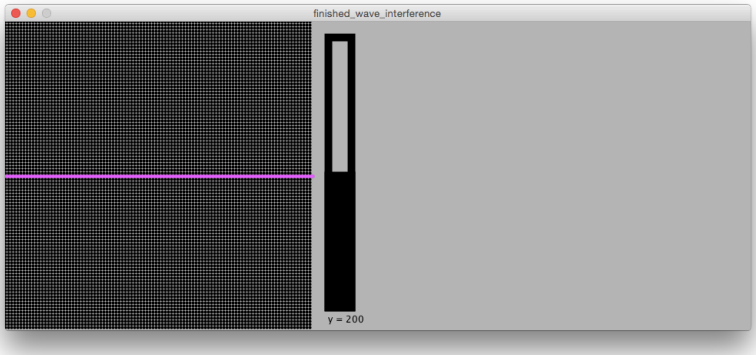


図 3.2: プログラム起動時の画面.

図 3.3 は 1 つの波源を生成した後, 周期をスライダーによって変更した様子である.

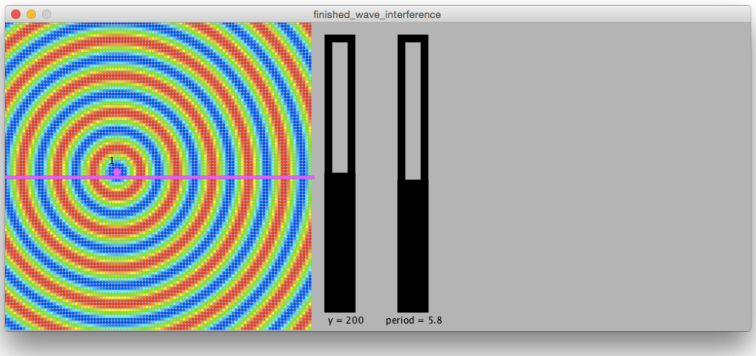


図 3.3: 1 つの波の周期をスライダーで変更した画面.

スライダーが周期を変更できる状態で L キー (lambda の頭文字) を押すと, 図 3.4 のように波の波長を変更できるスライダーに変化する. 周期を変更するスライダーに戻したい場合は P キー (period の頭文字) を押せばよい.

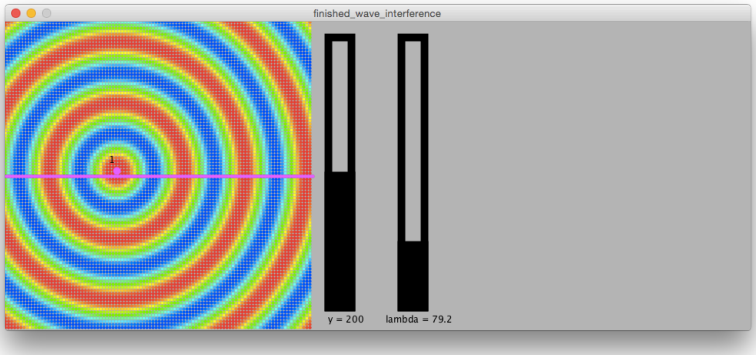


図 3.4: 波長を変更できるスライダーに変化させた時の画面.

波源は図 3.5 のように 5 個まで生成できる.

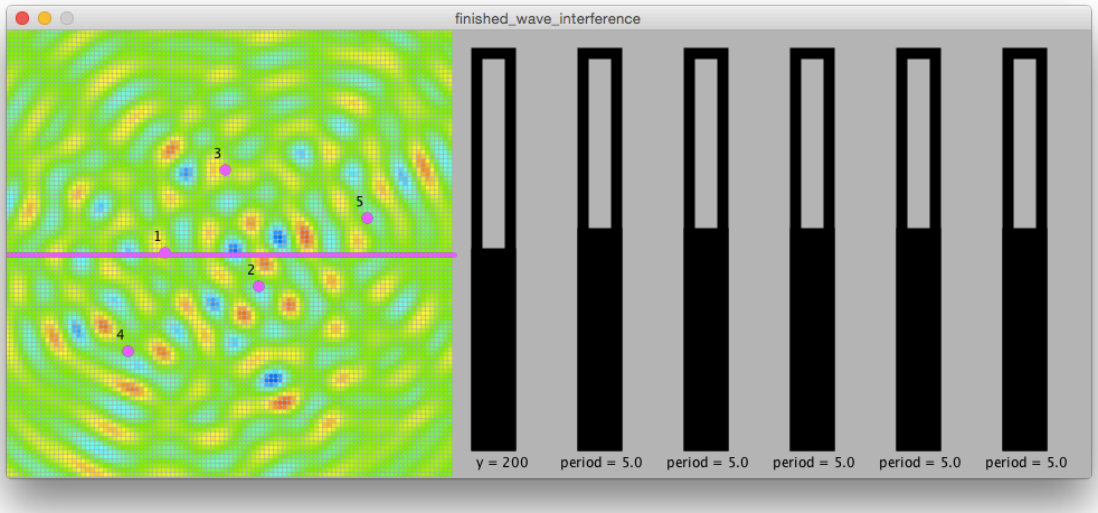
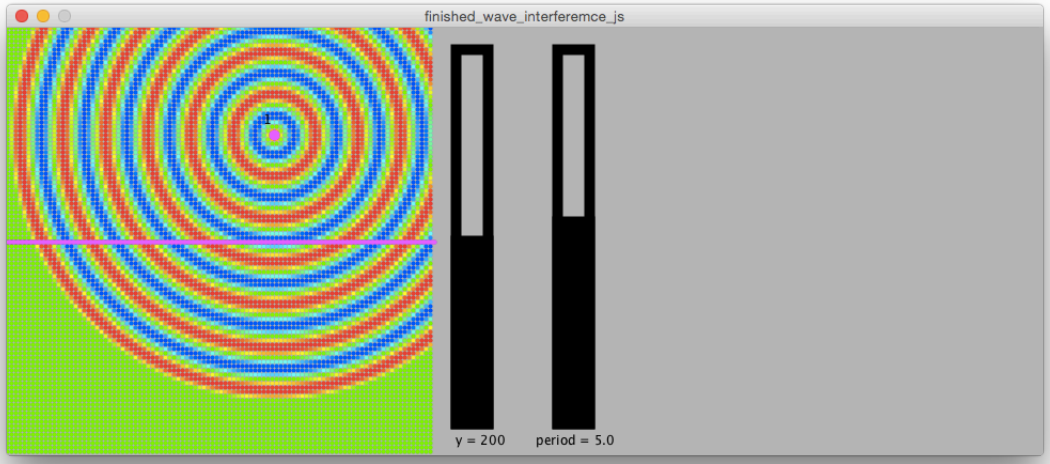


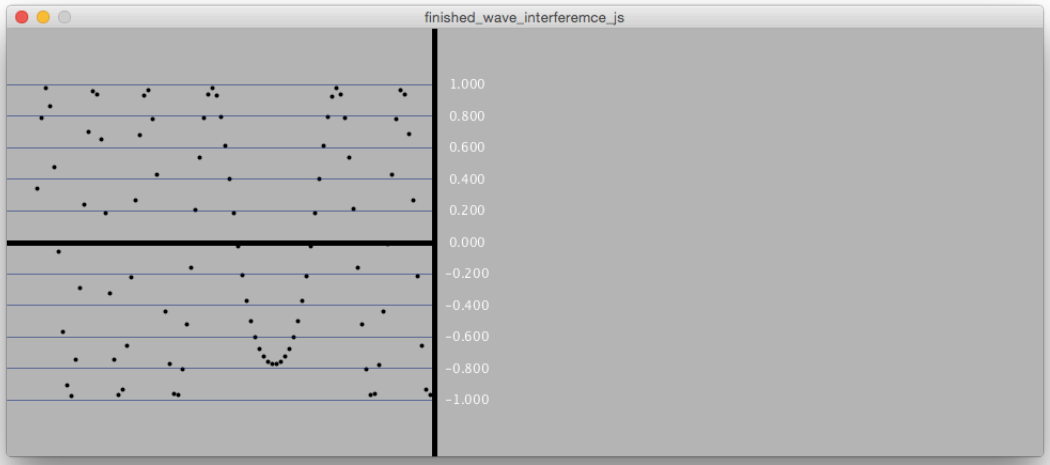
図 3.5: 波源を 5 個生成した時の画面.

3.1.2 check モード

このモードは draw モードに描写されている赤い線上の変位をリアルタイムに視覚化するモードである. 同時刻, 同座標で周期, 波長が同一な波を生成し, 波を生成して 360 フレーム目の状態を draw モード,check モードでそれぞれ描写したのが図 3.6(a),(b) である.



(a) draw モード.



(b) check モード.

図 3.6: 周期 5.0, 波長 40.0 の波が生成されてから 360 フレーム目の draw モード,check モード.

3.2　波の回折現象の視覚化

図 3.7 は障害物の間に一定の間隔で配置された波源から生成される円形波によって, 波の回折現象の視覚化を行うプログラムである.

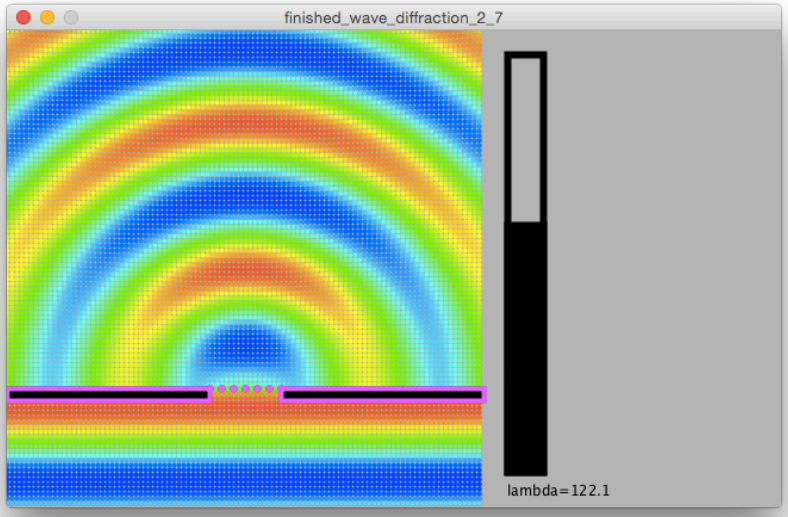


図 3.7: 回折現象の視覚化.

プログラムを起動すると図 3.8 の画面になる. この画面では画面右にあるスライダーで, 障害物の隙間の間隔を調節できる. 長さの値はスライダー下部に表示されており, 図 3.9 のように長さの値に応じて波源の数が決定される.

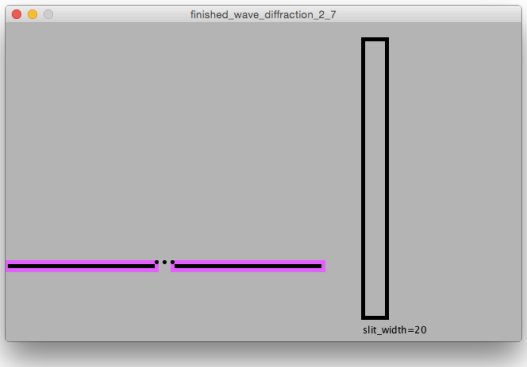


図 3.8: 初期状態

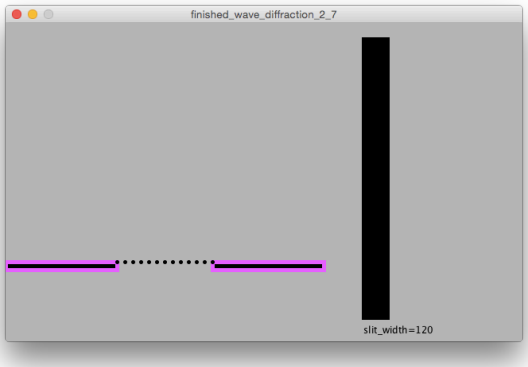


図 3.9: 隙間の間隔を 120 に設定した様子

これらの画面で S キー (Start の頭文字) を押すと, 図 3.10 の画面へと切り替わる. 図 3.10 の画面は画面下部から平行に進行してきた入射波が波源に到達すると, 図 3.11 のように円形波が生成され, 波長に応じた挙動を描写する.

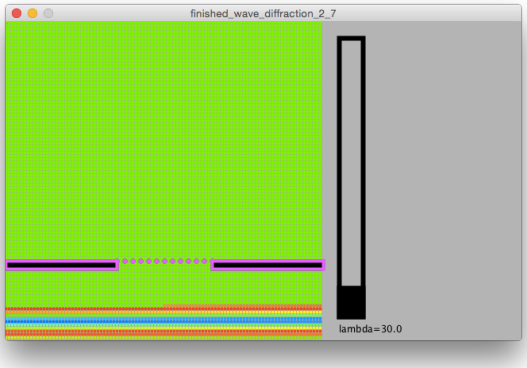


図 3.10: 入射波の描写

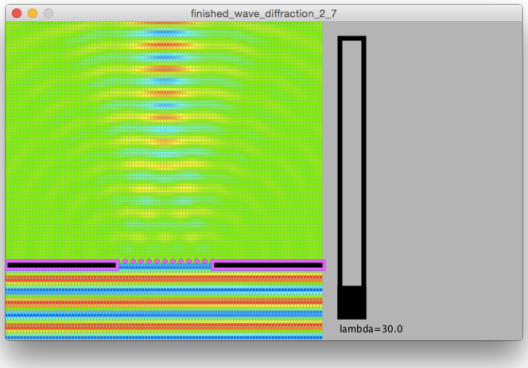


図 3.11: 隙間の間隔=120, 波長=30.0

波長を変えると図 3.12 のように, 図 3.12 と同じ波長の値のまま隙間の間隔を小さくしてシミュレーションすると図 3.13 のようになる.

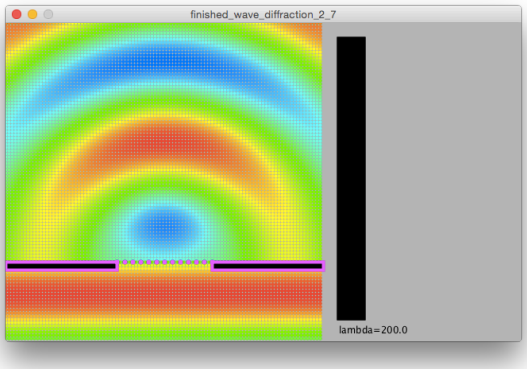


図 3.12: 隙間の間隔=120, 波長=200.0

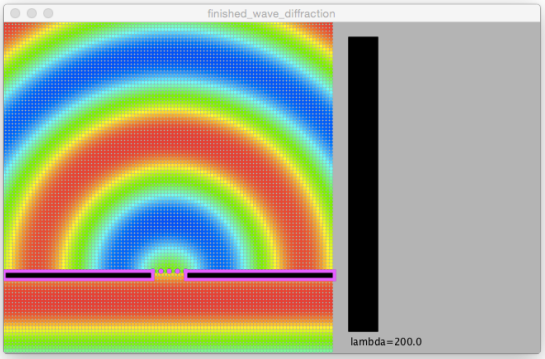


図 3.13: 隙間の間隔=40, 波長=200.0

第4章　プログラムの制作過程と解説

この章では今回作成したプログラムのうち, 波の干渉と回折現象を視覚化したプログラムの制作過程の記述と処理の解説を行う.

4.1　Processing 言語

本研究のプログラムに使用した言語は Processing 言語である. Processing 言語は以下に挙げるような特徴を有している [4].

1. 基本文法は Java をベースに記法を簡単化したものであり, インタラクティブソフトウェアやビジュアルプレゼンテーションを容易に実現することに特化している.

2. Windows, iOS, Android のタブレット端末で用いられる 3 つのプラットフォーム全てで動作する.

1 の特徴からは, 視覚化プログラムの作成に適していると言える. 2 の特徴からは, 学習者があらゆるタブレット端末を使用してもプログラムが動作することが言える. これらの理由から, 本研究で使用するプログラミング言語に最も適していると考えた.

4.2　波の干渉の描写

回折, 反射, 屈折の性質を可視化するためには多数の波を生成し, それらに重ね合わせの原理を適用しなければならない. そこで複数の点源から波を生成した上で, 重ね合わせの原理によって生じる干渉を視覚化するプログラムを作成した.

4.2.1　波源から波の代わりとなる円を描写するプログラム

最初は波の干渉の描写を実現するために, 図 4.1 のように, 画面左上から右下へ進行する斜めの線を入射波に見立て, 入射波が波源として設定した座標を通過すると, 波源から波の代わりとなる円を描写するプログラムを作成した. この後, 円が重なった部分の色を変化させたり複数の円の包絡線を太く描写することを考えた. しかし計算アルゴリズムが非常に複雑になることや, 処理速度が追いつかないことからこの方法は断念した.

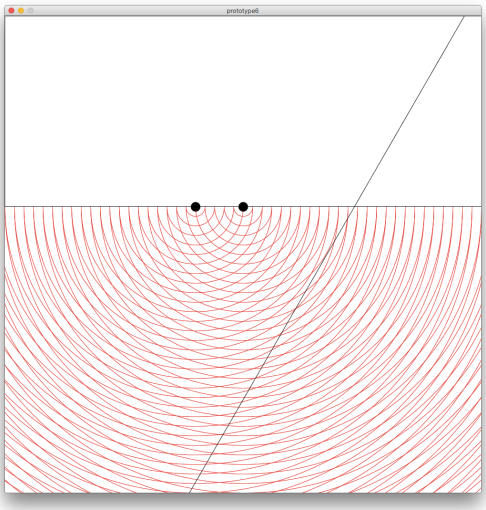


図 4.1: プログラムの動作画面.

そこで各ピクセルが波源からの距離, 波源の生成された時間, 波源から生成される波の波長と周期を基にして自らの地点の位相を計算する手法を考案した.

4.2.2　平面波描写のアルゴリズム

アルゴリズムは以下の手順である.

1. 平面波の変位計算を何ピクセルごとに行うかを指定する. これにより描写領域のピクセル数を擬似的に変更する. これ以降, 説明のために擬似的に変更したピクセルの単位を**擬似ピクセル**と記す.

2. 描写領域がクリックされると波源の座標情報が追加される. この際 1 つの波源ごとに, 各擬似ピクセルとの距離を計算し, 各擬似ピクセルに対応した配列 (point_distance[i][j])

に格納している.

3. 点源とは異なる座標の点における変位の計算を, 全ての点源と全ての擬似ピクセルに対しおこない, これによって得られた変位の値を, それぞれの擬似ピクセル上での変位に変換し, 各擬似ピクセルごとに設けた配列 (point_[i][j]) に足し合わせていく.

4. 各擬似ピクセルごとの変位の値を基にして, 値に応じた色の点を描写する.

5. 描画領域がクリックされた瞬間のみ手順 2 を, そうでなければ手順 3-4 を毎フレームごとに繰り返して平面波の挙動を継続的に描写し続ける.

なお, 手順 2 の段階で配列 (point_distance[i][j]) に値を格納しているのは, 手順 3 での異なる座標の点における変位の計算をおこなう際, 点源と擬似ピクセルの距離が必要であるため, あらかじめ計算させることで処理を軽くするためである.

4.2.3 描画領域のピクセル数を擬似的に変更する方法

4.2.2 に描写領域のピクセル数を擬似的に変更するとあるが, これを行う事により平面波描写の処理を大幅に軽減することができる.

まず, 図 4.2 のように 8 × 8 ピクセルの描写領域があるとする. これに対し 4.2.2 のアルゴリズムを適用すると, フレームごとに 8 × 8 の 64 個の座標それぞれに対し変位計算を行うことになる. 一方, 図 4.3 のように 8 × 8 の描写領域に対し, 変位計算を 2 ピクセルごとに行うとするならば, フレームごとの位相計算は 4 × 4 の 16 個の座標に対してすればよい. つまり計算回数は図 4.2 の時と比べ,4 分の 1 回となる.

今回作成したプログラムは描写領域を 400 × 400 ピクセルとしているため, ピクセル数を変更しなければフレームごとに 160000 回の変位計算を行うことになるが, 4 ピクセルごとに変位計算を行うことで計算回数を 10000 回まで削減した.

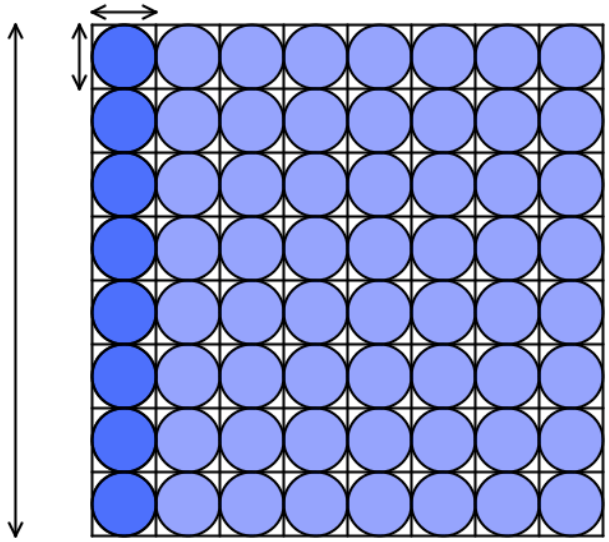


図 4.2: 8× 8 の描写領域に色を塗る場合.

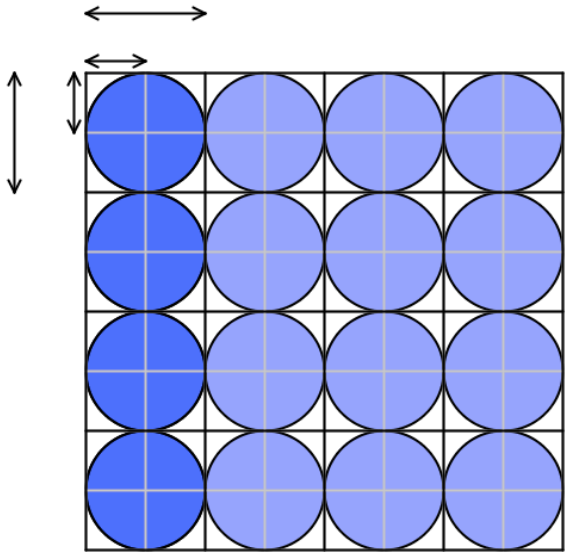


図 4.3: 8× 8 の描写領域を擬似的に 4× 4 に変更する場合.

またプログラムの擬似ピクセル数はグローバル変数の point_regulation の値を変えることで変更可能である. 図 4.4 は擬似ピクセル数を 1(変更なし) にした時の波の描写, 図 4.5 は擬似ピクセル数を 8 にした時の波の描写である.

なお, 図 4.4, 図 4.5 以外で論文に用いているプログラムの動作画面は全て擬似ピクセル数を 4 で設定したものである.

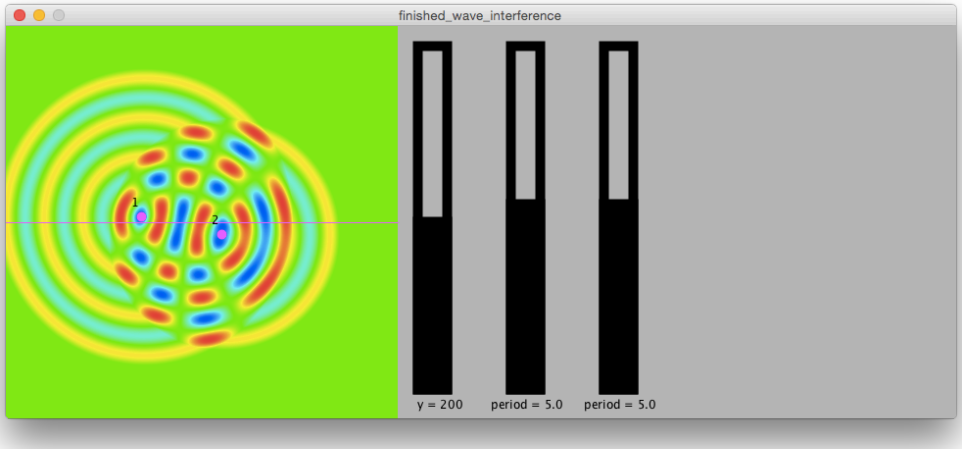


図 4.4: 擬似ピクセル数を 1 に設定した時の波の描写.

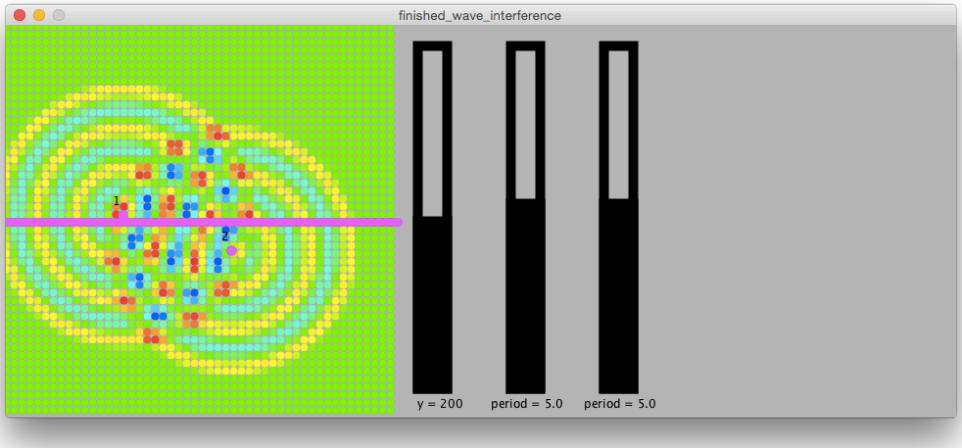


図 4.5: 画面の画素数を 8 に設定した時の波の描写.

4.2.4　波の色の表現方法

Processing 言語では色を表現する際,RGB カラーモデルと HSB カラーモデルの 2 つを用途に応じて使用することができる. RGB カラーモデルは赤 (Red), 緑 (Green), 青 (Blue) の 3 つの色を様々な配分で混ぜ合わせることで色を表現し, HSB カラーモデルは色相 (Hue), 彩度 (Saturation), 明度 (Brightness) の組み合わせによって色を表現する.

図 4.6 は HSB モードで S(彩度)=100,B(明度)=100 に設定し,H(色相) を変更した際の色の変化である. 図の右側の数字は H(色相) の値を示している. このように HSB カラーモデルでは RGB カラーモデルとは異なり, 赤, 緑, 青の 3 色を 1 つの値 (色相) を入れ替えるだけで表現できる特徴があるため, プログラムではこのモデルを採用した.

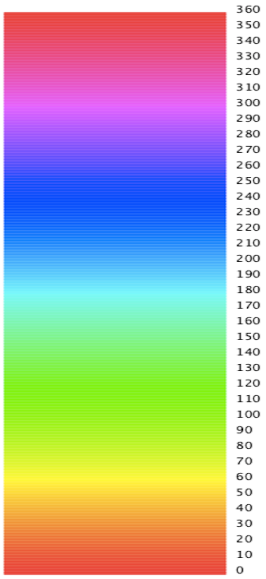


図 4.6: HSB モードで色相の値を上下させた時の色の変化.

プログラムでは, 擬似ピクセルごとに記録された波の変位に応じて描画する点の色を変更する. この時, 波の変位が高い時は赤色, 変位が 0 に近い場合は緑色, 変位が低い場合には青色で描写するように処理を行う.

```
point[i][j] += number_wave_point;
stroke( (number_wave_point2- point[i][j])*(230/number_wave_point2),100,100);
```

この処理は各擬似ピクセルごとの (変位) を 0∼230 までの値に変換するものである.

まず (point[i][j]) に格納されている擬似ピクセルの変位に現在の波源の数を足す. 例えば点源が 3 つある場合, 各擬似ピクセルの変位は波源の振幅が 1 であるため-3∼3 までの値をとる. ここに点源の数を足すと (point[i][j]) の値は 0∼6 までの値をとり, 全て正の数となる. 次に変位の値が大きいほど赤色に近づけるようにするため, 点源の数の 2 倍 (number_wave_point2) から (point[i][j]) を減算する. 最後に点源の数の 2 倍で割ることで値を 0∼1 の範囲にしてから 230 を掛けることで, 変位の変化と色の変化を連動させることができる.

4.3 Elementary_waves クラス

このクラスは波源と, 波源から生じる波に関する処理を行っている.

```
class Elementary_waves{
  float center_x; float center_y;
  float lambda;
  float period;
  float created_time;
  float period_max; float period_min;
  float lambda_max; float lambda_min;

  Elementary_waves(float _center_x, float _center_y,
  float _lambda, float _period,
    float _period_min, float _period_max,
    float _lambda_min, float _lambda_max){
     center_x = _center_x; center_y = _center_y;
     lambda = _lambda;
     period = _period;
     period_max = _period_max; period_min = _period_min;
     lambda_max = _lambda_max; lambda_min = _lambda_min;
     created_time = 0;
    }
```

```
    float now_displacement_y(float distance, float lambda,
    float period, float time_adjustment){
    float time_phase_adjustment;
    time_phase_adjustment =
    ((millis()-time_adjustment)/1000.0)*(360.0 / period);
    float distance_lambda_adjustment =
    (distance%lambda)/lambda*(360.0);
    float y =
    sin(radians(time_phase_adjustment -
    distance_lambda_adjustment));
    return y;
  }
}
}
```

4.3.1 メンバ変数

クラス Elementary_waves は波源が持つ様々なパラメータをメンバ変数としている. メンバ変数を以下に示す.

center_x, center_y

波源の x 座標,y 座標を記録する.

lambda

波源から生じる波の波長を記録する.

period

波源から生じる波の波長を記録する.

period_max, period_min

スライダーで周期を調節する際の最大値, 最小値を記録する.

lambda_max,lambda_min

スライダーで波長を調節する際の最大値, 最小値を記録する.

created_time

波源が生成された時間を記録する.

プログラムではクラス配列としてクラス Elementary_waves のインスタンスを複数生成し, 各波源としている.

4.3.2 点源とは異なる座標の点における変位の計算の実装

2.3 の方法で任意の擬似ピクセル上の変位計算を行うためのメンバ関数,now_displacement_y を実装した.

引数の distance にはある波源と, ある擬似ピクセルとの距離を代入する. lambda にはある波源から生成される波の波長,period にはある波源から生成される波の周期を代入する. time_adjustment にはある波源が生成された際に, プログラムを起動してから経過していた時間を代入する.

Processing 言語にはプログラムが起動してからのミリ秒 (1/1000 秒) の数を返り値とする millis() という関数が備えられている. 計算上ミリ秒ではなく秒数に直した方が都合がいいので,millis の値を 1000 で割っている.

また sin 関数の中の値を radians() 関数によって角度の単位を「度」からラジアンに変換している.Processing 言語のパラメータの単位はラジアンなのでこのような処理を施している.

4.4 指定した y 座標における x 座標の変位の描写

3.1.2 節で説明した,y 軸上の変位を描写するアルゴリズムを以下に示す.

1. 各擬似ピクセルの変位を計算する.
2. 手順 1 によって得られた値を擬似ピクセルごとに設けられた配列に減算して格納する.
3. draw モードで赤線が描写されている座標の変位を取得し, 振幅となる値を掛けた上で点として描写する.
4. 変位を測る目盛りを描写する.

5. 1-4 を繰り返す.

手順 2 で減算する理由は,Processing 言語で用いられる座標系を, 数学等で用いられる一般的な座標系に変換して描写するためである.

4.5 Slider クラス

波の干渉の視覚化プログラム内で, 榊の作成したプログラムを元に, 波源の数をいくら増やした場合でもスライダーの数がそれに応じて増えるようにクラス化した [3]. コードを以下に記す.

```
class Sliders{
  int position_x; int position_y;
  int slider_height; int slider_width;
  int digit;
  int number_tickmarks;
  int num_separator;
  float min; float max;
  float cordinate_y;
  String variable_name;
  boolean draw_flag;
  boolean slider_dragged;

  Sliders(int _position_x, int _position_y, int _slider_height,
   int _slider_width, float _min, float _max,
    int _number_tickmarks,int _digit, String _variable_name){
    position_x = _position_x; position_y = _position_y;
    slider_height = _slider_height; slider_width = _slider_width;
    min = _min; max = _max;
    number_tickmarks = _number_tickmarks;
    digit = _digit;
    variable_name = _variable_name;
    draw_flag = false;
```

```
    slider_dragged = false;
  }

  float draw_slider(float variable_value){
    if(draw_flag == true){
      num_separator =
       (int)map(variable_value, min, max, number_tickmarks-1,0);
      cordinate_y =
      position_y + slider_height -
      ((float)slider_height / (number_tickmarks-1))*num_separator;
      fill(255);
      noStroke();
      rect(position_x, position_y + slider_height, slider_width+50, 30);
      stroke(1);
      rect(position_x, position_y, slider_width, slider_height);
      fill(0);
      rect(position_x, cordinate_y, slider_width,
      position_y + slider_height - cordinate_y);
      if(variable_name == "y = "){text(variable_name +
      nf(variable_value*point_regulation,1,digit),
       position_x, position_y + slider_height + 20);
      }else{
        text(variable_name + nf(variable_value,1,digit),
         position_x-20, position_y + slider_height + 20);
      }
      if(mousePressed==false) slider_dragged=false;
      if(mouseX >= position_x & mouseX <= position_x + slider_width
      & mouseY<=cordinate_y+10 & mouseY>= cordinate_y-10){
        if(mousePressed){
          slider_dragged= true;
        }
      }

if(slider_dragged){
```

```
        num_separator+=
        (int)((cordinate_y - mouseY)/
        ((float)slider_height /(number_tickmarks-1)));
        num_separator =
        constrain(num_separator, 0, number_tickmarks-1);
        if(digit == 0){
          variable_value =
          (max-min) - round(map(num_separator, 0,
          number_tickmarks-1, min, max));
        }else{
          variable_value =
          (max+min) -
          map(num_separator, 0, number_tickmarks-1, min, max);
        }
        return variable_value;
      }else{
        return variable_value;
      }
    }else{
      return variable_value;
    }
  }
}
```

このプログラムを利用する準備として, 擬似ピクセル数を格納する変数 point_regulation をグローバル変数で定義しておく必要がある.

4.5.1 メンバ変数

クラス Sliders 内の変数を以下に示す.

position_x, potision_y

スライダー左上の x 座標,y 座標を記録する.

slider_height,slider_width

縦横の幅を記録する.

digit

返り値の桁数を記録する.

min,max

値の最小値, 最大値を記録する.

number_tickmarks

返す値を何段階で調節できるようにするかを記録する. 例えば min=0,max=10 の時,number_tickmarks を 11 に設定すると 0,1,2...9.10 の 11 段階で値を調節できるスライダーとなる.

num_separator

調節する変数の値がスライダーの何段階目の値に位置しているかを記録する.

cordinate_y

調節している変数の値に応じた y 座標を記録する.

variable_name

スライダー下部に表示される, 調節する変数名の文字列を記録する.

draw_flag

スライダーを画面に表示するかどうかの判定で利用する.

slider_dragged

スライダーの掴み判定の判定で利用する.

variable_value(draw_slider の引数)

調節したい変数を格納している.

4.5.2 処理の流れ

1. 調整する変数の値から, その値が下から何段階目に位置するかを計算する

2. 段階の番号から, スライダー部分の y 座標を計算する.

3. スライダーを描画する.

4. スライダーの掴み判定 (slider_drugged) を行う.

5. 掴み判定が true の場合,

   (a) マウスの移動距離に対応した段階数に値を加える.

   (b) 段階数から調節する変数の値を変更し, その値を返す.

   掴み判定が false の場合,

   (a) 調節する変数の値を変更せずにそのまま返す.

4.6 回折現象の視覚化

波の干渉プログラムで作成した平面波の描写をベースとして作成した. 波源の x 座標の間隔は, 障害物の間隔に関わらず 10 ピクセルとしている. 実際の物理現象を考えると波源は無限に存在するので波源同士の間隔は存在しないと言えるが, 仮に有限であったとしたら波源同士に間隔の差はないと考えたからである.

第5章　考察

この章では, 本研究で完成させることができなかった反射, 屈折の法則の視覚化プログラムについて, 現状の実装を記した上で, 処理の問題点を考察する.

5.1　反射の法則の視覚化

図 5.1 は反射面に一定の間隔で配置された波源から生成される円形波の変位情報から反射角を描写するプログラムである.

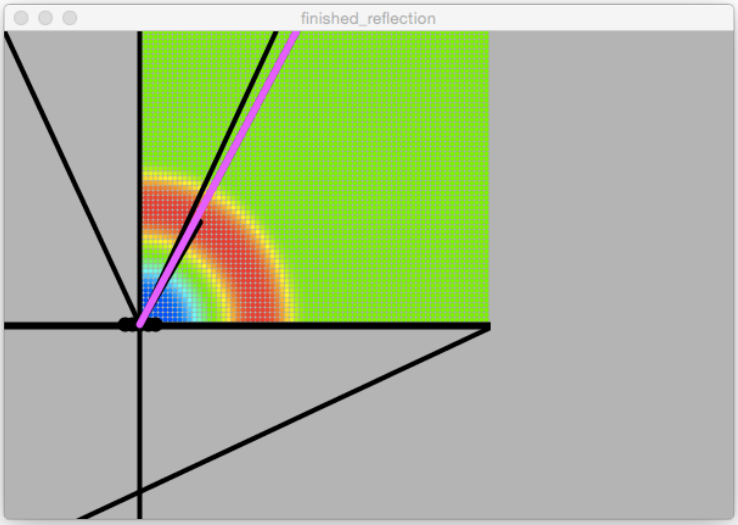


図 5.1: プログラムの動作の様子.

プログラムを起動すると図 5.2 のように入射波となる平行波が画面左上から右下にかけて進んでくる. 波源は入射波の速度に応じた間隔で 5 つ生成される.

入射角が 20 度の入射波によって各波源が生成され, それらから生じた波を描写したものが図 5.3 である. 図 5.3 の画面右上の波の描写領域に注目すると 3 本の線がある.

画面の上端に届いていない黒線は,3 個目の波源 (入射角と反射角の境界線) から現在の最大変位を結んだものである. 各フレームごとに最大変位が変わるため, この線もそれに応じて更新される.

赤線は各フレームごとの最大変位をとる座標の傾きを平均化して描写したものである.5.4.1 節で詳しく取り扱うが, プログラムの仕様上, 理論上の最大変位を取得することは出来ないので値を平均化することで精度を高める目的がある.

もう一つの黒線は, 反射の法則を満たす角度で 3 個目の波源から引いた線である. この線と赤線のずれを無くすことを目標としていた.

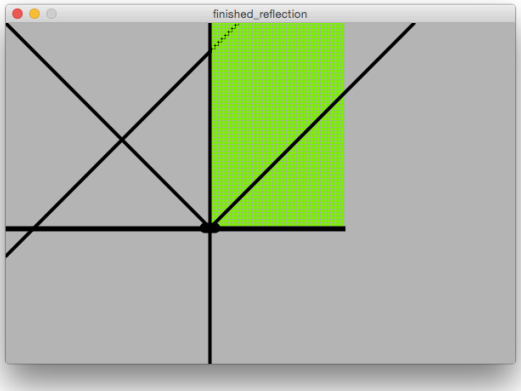


図 5.2: 入射波が進行する様子

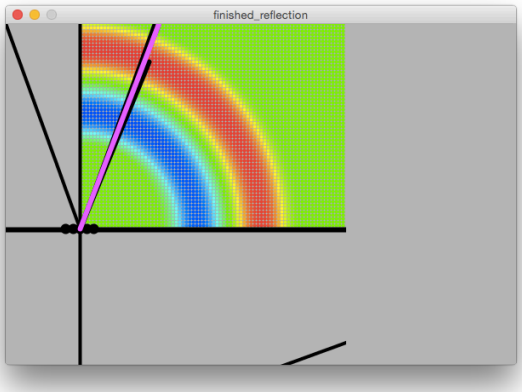


図 5.3: 入射角を 20° に設定した波が進行する様子.

入射波を 20° に設定した結果が図 5.4, 40° に設定した結果が図 5.5 である.

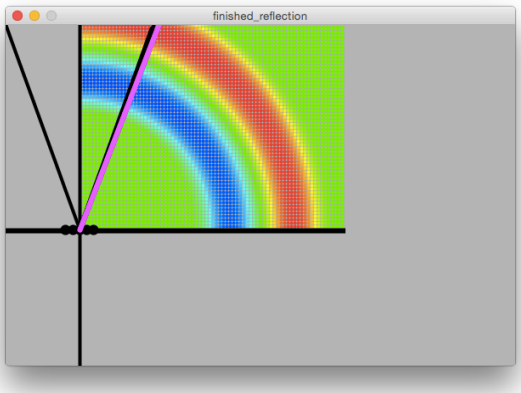


図 5.4: 入射角が 20° の時の反射角の結果.

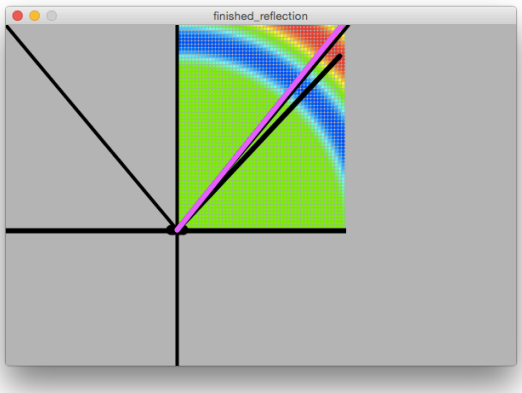


図 5.5: 入射角が 40° の時の反射角の結果.

図 5.4, 図 5.5 を見ると僅かながらずれが生じており, 本研究ではこのずれを修正することが出来なかった.

5.2　屈折の法則の視覚化

図 5.6 は屈折率が 1.5 の媒質に, 入射角 30° の入射波が進行した際の屈折角の角度を示すプログラムである.

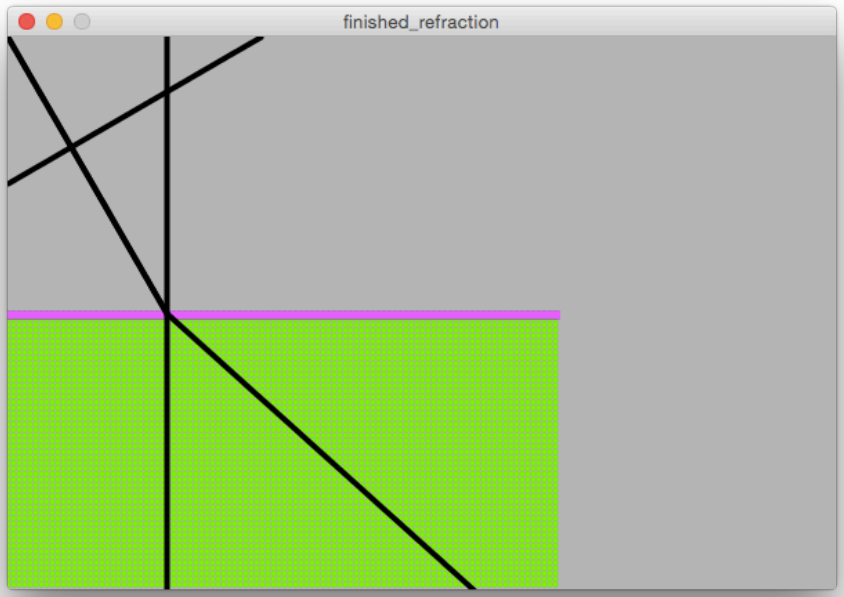


図 5.6: 入射角に対する反射角の角度.

この後, 媒質に応じた速度の素元波を発生させ,5.1 節で用いた方法と同じ要領で屈折角を描写しようと考えていたが, 反射角の描写が理論通りにできていないので, 処理の実装に取り掛かることができなかった.

5.3　最大変位の点へ線を引く理由

5.1 節で示したように, 本研究では, 波の最大変位を辿れば反射角, 屈折角の角度となると考えてプログラムを構築した. この考えに至った理由を以下に示す.

当初, 波の反射波, 屈折波をリアルタイムで描写する方法が思いつかなかったため, 入射波の角度に対して素元波がどのように広がっていくのかを描写するプログラムを作成し, 糸口を探った. 入射角を 40° に設定したものが図 5.7,50° に設定したものが図 5.8 である.

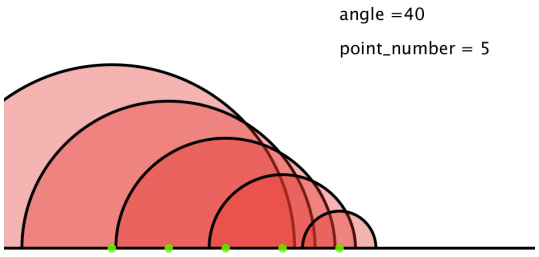


図 5.7: 入射波 = 40°

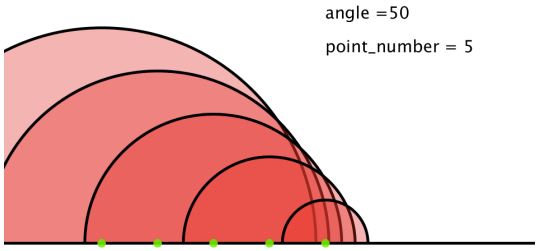


図 5.8: 入射角 = 50°

また図 5.8 の状態から一定時間経過した素元波を描写した図 5.9 を作成した.

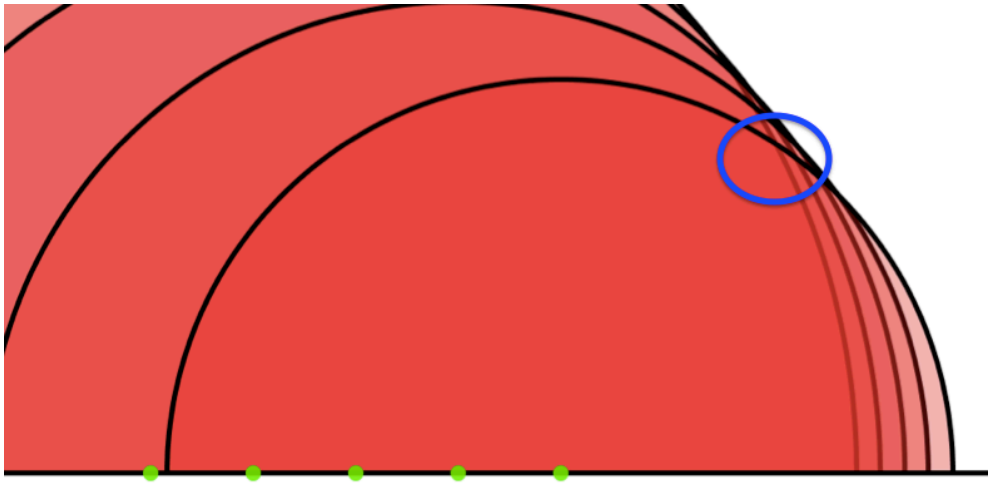


図 5.9: 図 5.8 から一定時間が経過した時の素元波.

図 5.9 の最大位相は青枠周辺だと考え, 青枠と中心の波源を結んだところ, 理想的な角度に近くなった. よって毎フレームごとに擬似ピクセルの中から最も変位の値が大きいものを選び出し, そこに対して線を引くことで反射角, 屈折角を表現すれば良いと考えた.

5.4　アルゴリズムの問題点

5.1 節で示したように理論上の反射角と 5.3 のアルゴリズムによって描写した反射角には若干のずれが生じる. そこでこの問題が生じる原因として 2 つの仮説を立てた.

5.4.1　擬似ピクセルの問題

プログラムの実装上, 擬似ピクセルの最小値は 1 であり, かつ整数値に限る. しかし本来の物理現象を考えると, 波の最大変位の座標は整数であるとは限らない. 図 5.10 の点 A が理論上の最大変位点とすると, 点 A に線を引くことはできず, 青点で示す各擬似ピクセルの左上の座標のいずれかにしか引くことができない. このずれが毎フレームごとに蓄積された結果, 適切な結果が得られなかったと考えられる.

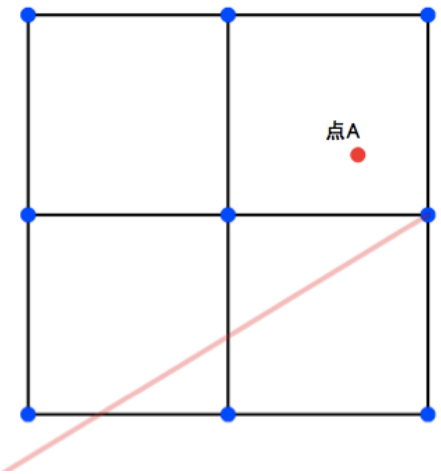


図 5.10: 理論上の最大変位とプログラム上の最大変位の誤差.

5.4.2　波源の個数の問題

本研究で作成したプログラムは波源の数を 5 個に設定しているが, 実際の物理現象を考えると波源は無限に存在する. 波源が無限に存在する故に波が反射や屈折を起こす際, 素元波の共通接線と垂直に交わる線が新しい波面となる理論が成り立つが, プログラムでは波源の数は有限なのでこれが完璧には成り立たない. これが値のずれを生んでいるのではないかと考えられる.

第6章　総括

本研究では, 複数の波源から生成される波が重ね合わせの原理により干渉する様子, 回折現象, 反射の法則, 屈折の法則の視覚化プログラムの作成に取り組み, 前者 2 つを完成させた. それにより得られた成果を以下に示す.

1. 任意の座標, 時間に波源を生成し, 生じる円形波の干渉の様子を確認できることや, 波源ごとに波長, 周期を調整できることから, 波の挙動を直感的に理解しやすくなった.
2. 数式等では特に説明し辛い, 回折現象を視覚化することができた.
3. プログラムの JavaScript 化を行うことでタブレット上での動作が可能となり, 今後教育現場で利用できる可能性を持たせた.

今後の課題を以下に示す.

1. 反射角, 屈折角を正確に表示できるようにするため, 本研究のプログラムとは異なる反射波, 屈折波の描写方法を考案しなければならない.
2. 教材として利用するには, スライダーの形等, デザイン性に欠ける部分がありこれらの改善を行わなければならない.

謝辞

本研究を行うにあたり, 終始多大なるご指導, 御鞭撻をいただいた西谷滋人教授に対し, 深く御礼申し上げます. また, 本研究を進めるにあたり, 様々な助力, 知識の供給を頂きました西谷研究室の同輩, 先輩方に心から感謝の意を表します. 本当にありがとうございました.

参考文献

[1] 「ICT 活用授業による学力向上に関する総合的分析評価」, 清水康敬, 日本教育工学会論文誌 32(3), (2008), 293-303.

[2] 「教育の情報化ビジョン　～２１世紀にふさわしい学びと学校の創造を目指して～」, 文部科学省, http://www.mext.go.jp/b_menu/houdou/23/04/__icsFiles/afieldfile/2011/04/28/1305484_01_1.pdf, 34.

[3] 「物理現象視覚化ソフトのライブラリ構築 」, 榊原健, 関西学院大学理工学部 情報科学科 卒業論文 (2015), 31-32.

[4] 「Arduino/Processing を用いたシステム制御実験のラピッドプロトタイピング」, 石川将人, 北卓人, 大須賀公一, 自動制御連合講演会講演論文集, 53(0), (2010), 247.