量子表現のためのクリエイティブ・コーディングツールキットの開発

若月 泰生[1,a)]　脇田 玲[1]

概要：量子は，しばしばスピリチュアルな存在やアート対象として認識される．しかし，量子特有の振る舞いを科学的側面から可視化するツールキットは存在する一方で，芸術的な側面から可視化するツールキットは存在しない．そこで，本研究では量子特有の振る舞いをアートとして画像や映像表現に応用することができるツールキットを開発した．本ツールキットは Python のライブラリとして使用でき，様々なアプリケーションやゲームなどへの応用が期待できる．

キーワード：量子，クリエイティブ・コーディング，Python ライブラリ

1. はじめに

微視的な物理現象を扱う学問として，量子物理学がある．量子物理学は，現代の科学を以ってしても理解することが困難であり，科学者のみならず，多くの人にとって不思議な存在として認識されている．それ故，しばしば芸術的な対象とされる．[*1]

量子を題材に，芸術的側面から表現した作品例として，Björn Dahlem の「Lux and Lumen (Cosmic Web)」[1] がある．作者は，インタビューの中で，光は Lux と Lumen の二つの意味があり，光の本質である光子もまた，量子物理学的に粒子と波動の二つの意味を持つことについて言及している．また，Markos Kay の「Quantum Fluctuations」[4] は，量子揺らぎという現象を表現した映像作品である．作者自身が CERN（欧州原子核研究機構）の科学者へインタビューし，それを通じて制作した．

一方で，semiconductor の「Parting the Waves」という作品は，量子シミュレーションを背景に制作された映像作品である．しかし，量子を題材とする作品で，数学や物理学を用いた例は少ない．また，このような作品を制作する過程で，どのような数学や物理学を用いるかは作者によるところが大きく，その詳細はあまり公開されていない．そのため，量子物理学を専門としない表現者にとって，数学や物理学を背景に，量子に纏わる作品を制作することは困難である．

近年，クリエイティブ・コーディングと呼ばれる，表現のためのコンピュータプログラミングが注目されている．クリエイティブ・コーディングをする上で，様々な現象を表現するための数値計算ツールキットは数多く存在するが，量子表現のための数値計算ツールキットは存在しない．そこで，本研究では，量子の振る舞いを表現するためのクリエイティブ・コーディングツールキットを Python ライブラリとして開発した．

2. 関連研究と本研究の立ち位置

2.1 関連研究

本研究を行う上で，量子シミュレーションのアルゴリズムの開発は，科学的可視化ツールを参考にした．理由は，量子の表現に特化したツールキットの事例がほとんどないためである．そのため，関連研究においても，科学的側面から量子を可視化したシミュレーションツールを示す．

主に科学者や物理学を専門とする学生を対象にした量子可視化ソフトウェアに，QMBlender[3] や QMwebJS[2] がある．これらは，専門家や学生を補助するためのツールであり，表現の自由度は少なく，物理的現象を理解するための可視化に特化している．

GitHub で公開されている Python ライブラリとして，QMsolve[5] がある．シュレーディンガー方程式のソルバーの提供を目指したライブラリであり，動画による可視化が可能である．本研究は Python ライブラリでのツールキット開発を行なっているため，次節で本ツールキットと QMsolve[5] について比較を行う．

[1] 慶應義塾大学大学院 政策・メディア研究科
[a)] tw0609@sfc.keio.ac.jp

[*1] 2002 年，英国物理学会の会員誌「Physics World」の読者投票により，最も美しい実験として「二重スリット実験」が選ばれた．

2.2 本研究と関連研究の違い

本ツールキット（Quantum-Art-Library．以下，QAL と呼ぶ．）は，第 1 章でも言及した通り，芸術的側面と科学的側面の両方から量子を表現することを試みる．関連研究で示した QMsolve[5] は，科学的側面からの可視化をしている Python ライブラリである．QAL と QMsolve の具体的な違いは，表 1 に示す通りである．

表 1 QAL と QMsolve の比較

	QMsolve	QAL
ソルバー	複数	1 種類のみ
ポテンシャル（障害物）	設定不可	設定可
量子の初期位置	設定不可	設定可
量子の速度	設定不可	設定可
データの出力	出力不可	出力可
シェーダーの読み込み	不可	可
可視化	動画	画像と動画

QMsolve には，ソルバーが複数あり，1D，2D，3D の 3 つの次元での量子の現象をシミュレーションすることができる．一方，QAL は，2D のみで，かつ 1 つのソルバーで開発を行なった．理由として，1 つのソルバーで様々な芸術的側面からの表現方法を追求することができれば，シェーダーの読み込みや出力方法など，表現の自由度を高めるための機能は他のソルバーでも応用することができると考えたためである．

QMsolve の場合，ソルバーは多いが，一つ一つのソルバーでの自由度はほとんどない．例えば，ポテンシャル（本研究では量子の障害物と定義する）の設定では，QMsolve はソルバーを選んだ時点で予め設定されているが，QAL では，任意の画像を読み込むことで設定することができる．量子の初期位置や速度についても，QMsolve では予め設定されているのに対して，QAL では表現者が自由に設定することができる．

出力については，QMsolve は動画のみであるが，QAL では，画像と動画を出力することができる．さらに，画像や動画の出力の際にシェーダーを読み込むことができ，より自由な表現が可能である．また，他のツールへの接続の可能性も考慮し，シミュレーション結果のみを配列で出力することもできる．

3. QuantumArtLibrary（QAL）の仕組み

3.1 構成

QAL の大まかな構成は図 1 に示す通りである．まず，任意の画像を読み込み，その画像を cv2 ライブラリ（OpenCV）を用いてグレースケールに変更する．次に，グレースケールに変更された画像をもとに，量子の時間ごとの確率分布をシミュレーションする．シミュレーション結果は 0〜1 の範囲を取り，この結果を出力することが可能である．シミュレーションには，Numpy や Cupy といった数値計算ライブラリを用いる．また，シミュレーション結果から，pyOpenGL や glfw を用いて，連番画像と動画を出力することができる．

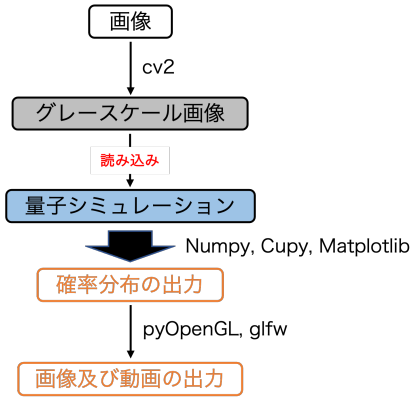

図 1 QAL の構成

3.1.1 開発言語

開発言語は主に Python を使用した．一部，シェーダー言語である GLSL を用いた．Python を用いた理由として，数値計算が容易にできるライブラリ（Numpy）や，NVIDIA の GPU を扱えるライブラリ（Cupy）が存在することが挙げられる．また，グラフィックスライブラリである，OpenGL や glfw も扱えることも Python を選んだ理由のひとつである．

3.1.2 使用ライブラリ

開発で使用した Python ライブラリとその役割は，表 2 に示す通りである．Cupy については，NVIDIA の GPU を使用して量子シミュレーションを行いたい場合のみ必要であり，QAL を使用する上で必須なライブラリではない．また，time や os といったライブラリは Python の標準ライブラリであり，別途インストールする必要はない．

表 2 開発に使用した Python ライブラリ

ライブラリ	役割
NumPy	量子シミュレーション（CPU 計算）
Cupy	量子シミュレーション（GPU 計算）
Matplotlib	カラーマップの適用
pyOpenGL	画像のレンダリング
glfw	画像のレンダリング
PIL	画像の書き出し
cv2	画像の読み込みと書き出し
progressbar	量子シミュレーションの進行状況を表示
time	量子シミュレーションの計算時間を表示
os	フォルダやファイルの処理

3.2 量子シミュレーションのアルゴリズム

基本的なシミュレーションアルゴリズムは，QMsolve[5]をベースに開発した．QMsolve と同様に，非線形シュレーディンガー方程式をスプリットステップフーリエ法で数値計算することで，量子の確率分布を求めた．ただし，数値計算に用いるポテンシャルは，QAL と QMsolve では異なる．

QAL，QMsolve 共に，シュレーディンガー方程式を数値計算する際に必要な変数として，ポテンシャルがある．図 2 は，QAL で二重スリット実験を出力した画像である．ポテンシャルとは，図 2 のうち，スリットのような障害物を指す．QMsolve では，二重スリットなどの予め設定されたポテンシャルしか用いることができないが，QAL では，読み込まれた画像からポテンシャルを設定することができる．

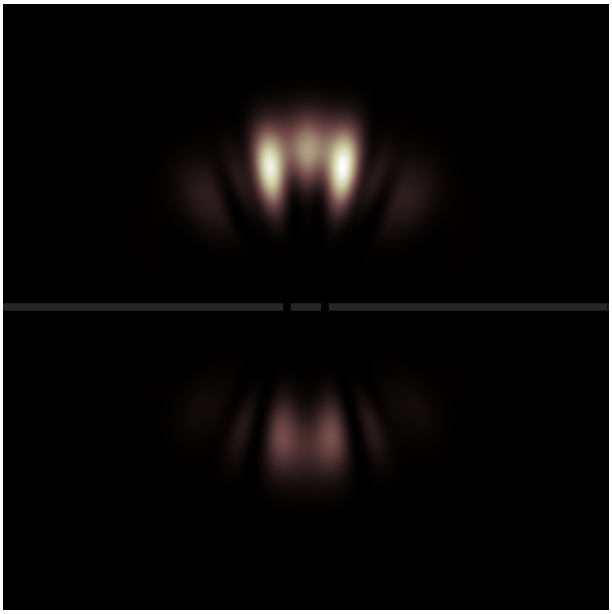

図 2 QAL の出力例（二重スリット実験）

また，量子をシミュレーションする際に，量子の初期位置や速度も必要な変数として挙げられる．QMsolve では，これらの値も予め決められた値であり，表現者が自由に値を決めることができない．一方で，QAL では，初期位置や速度も決めることができる．

その他に，解像度の設定やシェーダーの読み込み，読み込んだ画像の色の調整ができる変数など，表現の自由が広がるような設計にした．

3.3 出力

QAL は，量子のシミュレーション結果のみの出力と，画像及び動画の出力の二種類の出力方法がある．

3.3.1 シミュレーション結果の出力

量子シミュレーションは，任意の格子点数で実行することができる．何も指定しない場合，400 × 400 の格子点数でシミュレーションされる．確率分布は，0 ～ 1 の範囲で表され，1 に近いほど量子の存在確率が高い．確率分布の出力は，「格子点 × 格子点 × step 数」の 3 次元配列として出力することができる．step 数とは，量子シミュレーション全体の時間を何分割で行うかという値である（量子シミュレーション全体の時間は QAL で予め設定しているため，表現者による変更はできない）．何も指定しない場合は，300 として計算される．

3.3.2 画像及び動画の出力

シミュレーションにより得た 0 ～ 1 の確率分布を matplotlib のカラーマップを用いて表現し，連番画像と mp4 動画として出力できる．シミュレーションする際に指定した格子点数は，出力される画像や動画のピクセル数であり，step 数は連番画像数やフレーム数と言い換えることができる．何も指定していない場合，400 × 400 の解像度で，300 枚の連番画像と，10 秒間（30fps）の動画が出力できる．

4. QAL のパフォーマンス検証

本章では，PC の性能による QAL のパフォーマンスを検証する．

QAL では，NVIDIA の GPU を利用した量子シミュレーションも可能である．そのため，Intel の CPU を搭載した MacOS 機と，Intel の CPU と NVIDIA の GPU を搭載した Windows 機の 2 台の PC を用いて，合計 3 パターンで比較した．表 3 は，それら 3 パターンで量子シミュレーションした際のパフォーマンス比較の結果である．比較の条件は，画像の読み込みは行わず，二重スリット実験をシミュレーションし，格子点数は 400 × 400 で，300 フレームの出力とした．また，量子シミュレーションにかかる時間のみを計測し，画像や動画を出力する時間は含まない．計測は time ライブラリで，それぞれ 3 回ずつ行なった．計測単位は sec（秒）である．

表 3 CPU と GPU のパフォーマンス比較

	MacOS CPU	Windows CPU	Windows GPU
詳細	7th Gen Intel Core i5	11th Gen Intel Core i7	NVIDIA GeForce RTX 3060
1	206.01 sec	207.06 sec	12.87 sec
2	206.65 sec	206.66 sec	12.78 sec
3	203.68 sec	206.46 sec	12.76 sec
平均	205.45 sec	206.73 sec	12.80 sec

CPU 計算について，MacOS 機と Windows 機による違いはほとんどなかった．一方で，CPU と GPU による計算速度には大きな違いが生じた．表 3 から，GPU の方が約 16 倍早く計算できることが分かる．

5. QAL の使用方法と作品例

5.1 QAL の使用方法

QAL のソースコードは，GitHub で公開している[*2]．また，PyPI（Python Package Index）にも登録している．そのため，インストールは，pip コマンドで行うこともできる．

QAL には，量子シミュレーションの結果を出力する関数と，画像及び動画を出力する関数の 2 つの関数しか存在しない．表現の自由度を高めるため，2 つの関数の引数には多くの設定が可能だが，ほとんど何も設定しない場合でも実行することができる．

詳細な使い方やインストール方法等は，GitHub[*2]や PyPI[*3]を参照されたい．

本章では，QAL を用いて制作した作品例を 3 つ示す．

5.2 作品例 1

1 つ目は，二重スリット実験を表現した作品である（図 3）．QAL は，何も画像を読み込まない場合，二重スリット実験をシミュレーションするように設計している．格子点数は 800×800 で，900 枚の連番画像を出力した．また，シェーダーの読み込みは行なっていない．

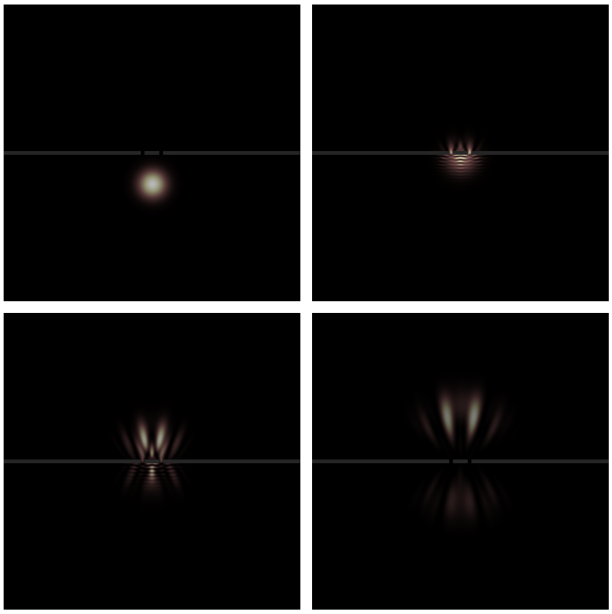

図 3 作品例 1

5.3 作品例 2

2 つ目は，図 4 に示した木のイラスト画像[*4]を読み込み，作品を制作した（図 5）．数式で表すことが困難な複雑な形を用いた量子シミュレーションは，画像を読み込むことができる QAL だからこそ可能である．そのため，図 4 のような樹木のイラストを読み込み，作品を制作した．この作品では，量子の初期位置と進む方向を設定することにより，画像の上から下へ量子をシミュレーションした．作品例 1 と同様に，格子点数は 800×800 で，900 枚の連番画像を出力し，シェーダーの読み込みは行なっていない．

図 4 木のイラスト画像

図 5 作品例 2

[*2] QuantumArtLibrary
`https://github.com/taikiwakatsuki/Quantum-Art-Library`

[*3] `https://pypi.org/project/QuantumArtLibrary/`

[*4] 著作権フリーな画像やイラストを提供するサイトである pixabay から引用.
`https://pixabay.com/ja/`

5.4 作品例 3

3 つ目は，キーボードの画像[*4]（図 6）を読み込み，作品を制作した（図 7）．図 6 のように，画像の色が特定の色に集中している場合，QAL では，グレースケールの画像に変換する際に，読み取る色の範囲を指定することで，少ない色の変化をシミュレーションに反映させることができる．そのため，この作品では，意図的に白を基調とした画像を読み込み，作品を制作した．グレースケールの画像に変換する際に，キーボードの影になっている黒い部分をあえて強調するような形で変数を設定した．作品例 1 と 2 と同様に，格子点数は，800 × 800 で，900 枚の連番画像を出力し，シェーダーの読み込みは行なっていない．

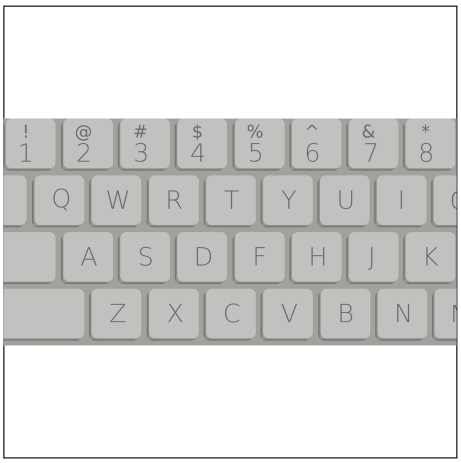

図 6 キーボードの画像

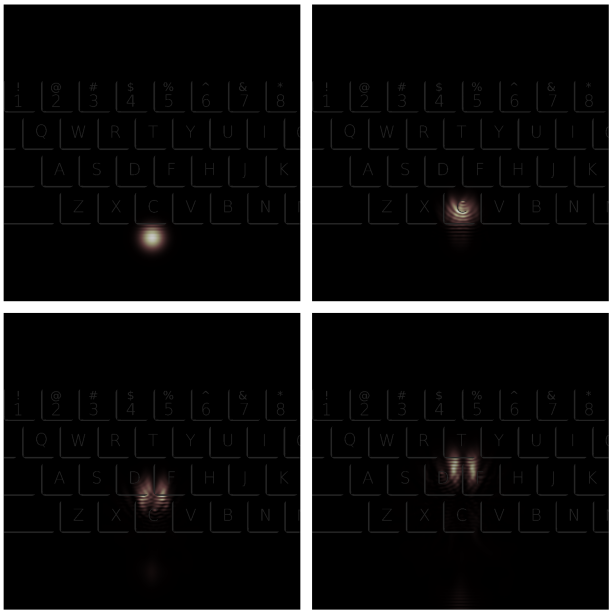

図 7 作品例 3

6. 結論

本研究では，量子表現のためのクリエイティブ・コーディングツールキットの開発を行なった．QAL では，科学的側面からの可視化ツールではできない，量子表現の機能を実現した．量子物理学を専門としない表現者でも数学や物理学を背景にした量子シミュレーションが可能であり，かつなるべく多くの変数を操作できるようにしたことで，自由度の高い表現が可能である．また，シェーダーを読み込むことができるため，シェーダーを扱える表現者にとっては，より自由な表現ができる設計となっている．

QAL に期待できる応用例として，他のアプリケーションやゲームなどへの接続が挙げられる．TouchDesigner や Blender など，Python を用いることができるソフトウェアは多く，量子シミュレーションの結果を他のソフトウェアへ用いることは比較的容易である．

今後の展開としては，より多くの量子表現をするため，ソルバーを増やしていきたいと考える．また，2 つの量子の操作や 3D での表現などの実現も目指していきたいと考えている．

参考文献

[1] Dahlem, B.: Lux and Lumen (Cosmic Web), `https://www.kiyoharu-art.com/?p=600` (2018). (Accessed on 07/30/2022).

[2] Figueiras, E., N. Olivieri, D., Paredes, A. and Michinel, H.: QMwebJS—An Open Source Software Tool to Visualize and Share Time-Evolving Three-Dimensional Wavefunctions, *Mathematics*, Vol. 8, No. 3, p. 430 (2020).

[3] Figueiras, E., Olivieri, D., Paredes, A. and Michinel, H.: QMBlender: Particle-based visualization of 3D quantum wave function dynamics, *Journal of Computational Science*, Vol. 35, pp. 44–56 (2019).

[4] Kay, M.: Quantum Fluctuations, `http://www.mrkism.com/quantum.html`. (Accessed on 07/30/2022).

[5] marl0ny and de la Fuente, R.: GitHub - quantum-visualizations/qmsolve: A module for solving and visualizing the Schrödinger equation., `https://github.com/quantum-visualizations/qmsolve`. (Accessed on 07/30/2022).