JumpLab: 2Dジャンプゲームを題材としたゲーム開発教材

福地 健太郎[1,a)]　伊藤 正佳[1]

概要：コンピュータゲーム開発において，キャラクターの見た目や動き，またその動かし方やカメラモーションなどのいわゆる 3C (Character, Controls, Camera) にまつわるパラメタを初期段階で確定させることはゲーム体験の質を向上させるために重要とされている．しかしそれらパラメタの調整でゲーム体験が大きく変化することを簡単に体験できる教材は少ない．そこで 2D ジャンプゲームを対象とパラメタ調整に主眼を置いた実験教材「JumpLab」を開発した．

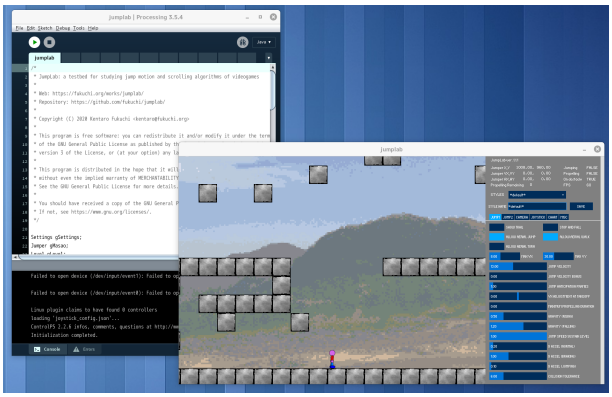

図 1　JumpLab 実行時のデスクトップ（部分）の様子．右側が主ウィンドウ．左側は Processing 言語の実行環境．

1. はじめに

コンピューターゲーム開発において，‘pre-production’ すなわち開発の初期段階においてゲームの中核となる部分を試行錯誤的に練り上げ，しっかりとそれを定めてから後の工程に移ることの重要性が広く指摘されている．中盤から終盤にかけての開発でこうした中核部分についての修正が入るとそれにあわせてレベルデザインを修正する必要が生じてしまい，開発工数の増大を招くためである．

pre-production において何を達成すべきか，何を重視すべきかについては様々に提唱されているが，ここでは，ゲーム開発者として，またゲームプロデューサーとして多数のゲームスタジオでのプロデュースおよびコンサルテーションの経験を持つ Mark Cerny が 2002 年の講演で述べた以下 4 点 [1] のうちの “3C” に注目する．

- 3C’s: character, camera, control
- game look
- key technology
- holistic game design (if appropriate)

Cerny は 3C, すなわちキャラクター・カメラ・コントロールを pre-production 期間で集中的に取り組むべきことの最上位に挙げている．ここで「キャラクター」は，主にプレイヤーが操作するメインキャラクターの見た目やその動き方を指す．「カメラ」はカメラの動きであり，これはプレイヤーキャラクターの周辺をどうプレイヤーに見せるかを通じて，ゲーム空間をプレイヤーにどう把握させるかをも意味する．「コントロール」はプレイヤーにどのようにキャラクターを操作させ，それを通じてゲームを楽しませるかを決定する要素である．Cerny はさらに，3C はキャラクターアクションゲームに特化した観点に見えるかもしれないが，同じことが他ジャンルのゲームにも適用できるとしている．

3C の重要性はゲーム業界において広く認識されている．例えばゲームデザイナーの Chris McEntee は Ubisoft Montpellier が開発した “Rayman Origins” の開発経緯を振り返った記事中で “At Ubisoft, the three Cs of a game (camera, character, and controls) are the most important thing to define and maintain because they determine the player’s experience throughout the entire game” と述べており，Ubisoft がゲーム体験を左右する最重要項目として 3C を捉えているとしている [4].

一方で，これを体系的に教える教材は，その重要性と比べると数が少ないように感じられる．ゲーム開発を教える教科書や書籍ではその多くがキャラクターの動きやその操作方法，またカメラモーションについてそれらの重要性を指摘してはいるが，それらをどのように実装するかについて教えることを主眼としており，その細かな調整過程を扱ったものは少ない．

[1] 明治大学
[a)] kentaro@fukuchi.org

その上で著者らが特に感じるのが，実際に手を動かしての 3C の試行錯誤を補助し，その重要性についての理解を促すインタラクティブ教材の必要性である．著者らが所属する大学の学部ではコンピューターゲームの開発を志す学生が少なくなく，その多くが独学でのゲームの自作に挑戦するが，一通り動くゲームを開発するところまでが精一杯で，その調整に時間をかけられる者は数える程しかいない．特に，上述した理由で 3C の調整を疎かにしていたためにバランス調整に苦労をしているとおぼしき例を多く見かける．

そこで我々は 2D の横スクロールジャンプアクションゲームを題材に，3C の調整，特にジャンプアクションとカメラモーションに特化した教材 “JumpLab” を作成した．JumpLab は Processing 言語で開発され，GNU General Public License 3.0 の元で公開されている[*1]．また 2021 年より JumpLab の Unity 版の開発を始めている．

本論文では JumpLab の設計方針および諸機能について報告する．なお JumpLab を用いての教育についてはまだ実践事例がないため，本論文では述べていない．

2. 背景

前章で述べたように，3C (Character, Camera, Controls) はゲーム開発の初期段階で入念に調整されるべきことが期待されている．前述の McEntee は次のように述べている [4].

> The player mechanics were designed and implemented iteratively, and the animations were changed and tweaked and thrown away until the three Cs felt perfect. Only once the main character was finished and comfortably fun to control did we begin working on level and gameplay element design.

同記事は Ubisoft Montpellier が開発した PlayStation 3 用ゲーム “Rayman Origins” の開発過程における preproduction の重要性について言及したものだが，その過程において 3C が “perfect” であると感じられるまで何度も作業が繰り返されたとしている．そしてそれが済んで初めて，レベル（いわゆる「面」「ステージ」のこと）やその他の要素のデザインに着手したとしている．

3C の重要性について説く文献において，では具体的に 3C をどのように設計すべきかについて明確な指針を示した例は少ない．おそらくは個々のゲームでどのようなゲーム体験を目指すかによって，到達すべき 3C の目標がそれぞれ異なることがその要因であろう．また，「ゲーム体験」そのものが，明確な定義を与えるのが困難という事情もある．関連語として ‘game feel’（ゲーム感覚，プレイ感覚）が挙げられるが，これについては Steve Swink が著書 [6] で，“real-time control of virtual objects in a simulated space, with interactions emphasised by polish” と定義を与えているが，やはりここから明確な設計指針を引き出せるようなものではない．

一方で，3C の個々の要素については様々な形で調査・研究が進められている．ここでは直接参考にしたものについて言及するにとどめる．Martin らは数種のゲームタイトルのジャンプ軌道を分析し，ジャンプ軌道の要素を 21 に分類している [2]．Summerville らはこの分類を参照しつつ，ファミリーコンピュータ (NES) 期のジャンプアクションゲーム (“platform game”) を題材に，それらのゲームにおけるジャンプの軌道をゲームのプレイ画面から自動で抽出する機構を開発し，種々のゲームのジャンプ軌道を可視化している [5]．ジャンプアクションゲーム含む横スクロールゲームのカメラモーションについては Karen が包括的な調査結果を報告している [3]．JumpLab で採用したパラメタ名はこれらの先行調査に倣った．

3. JumpLab 設計指針

3.1 用途および対象ユーザ

JumpLab は，ゲーム開発初期段階における 3C の重要性を教えるための教材としての用途を第一に設計している．具体的には，ジャンプアクションゲームを題材に，特にジャンプアクションおよびカメラモーションにまつわる種々の要素を実行時に調整可能としており，ジャンプ操作をしながら同時にパラメタ調整を繰り返し高速に行えるようにすることで，パラメタ調整によってゲーム感覚 (game feel) がいかに変化するかを学習者に体験させ，それを通じて 3C の重要性を意識してもらうことを目的としている．

対象ユーザは，第一には後述するように著者の一人が大学で受け持っている学部 2・3 年生向けの講義で使用することを目的としたため，ゲーム開発やインタラクション設計に関心を持つ大学生を対象としている．ただしツール自体は簡単な操作で使えるよう設計してあるため，対象ユーザは大学生には限定されない．本ツールを用いて行われる指導を理解するためには基本的な物理運動に関する知識が求められるため，実践的にはおそらくは中学生以上が対象となるだろう．

3.2 全体設計

JumpLab の画面構成は，**図 2** に示すようにゲームのプレイ画面と可変要素の調整用画面とを隣接させている．この状態で学習者は，キャラクターの操作と可変要素の調整とを交互に繰り返して，ゲーム感覚の変化を観察し 3C の調整について学ぶ，というのが基本的な使用形態である．このとき，プレイヤーキャラクターの操作はキーボードもしくはジョイスティックによって，可変要素の操作は主に

[*1] https://fukuchi.org/works/jumplab/ より取得可能.

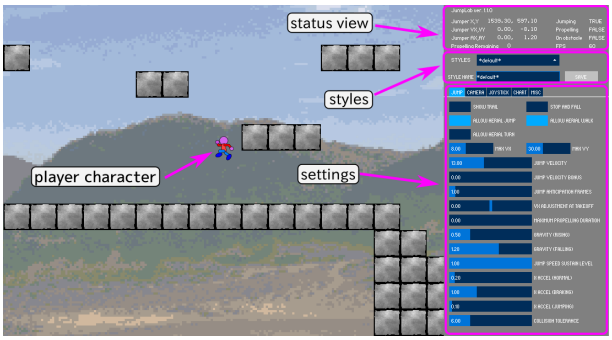

図 2 JumpLab の画面構成．プレイ画面と可変要素の調整用画面は常に表示されている．

マウス操作によって行う[*2]．

開発には Processing を用い，またキャラクターの制御や障害物との衝突判定などは既存の物理エンジンを用いずにすべてを書き起こした．これは本ツールがゲーム開発を志す者向けの学習用教材として設計されているため，いつでもソースコードエディタと実行環境とが一体化している Processing を用いることで，すぐにソースコードを参照することができ，また必要に応じてその改変を行いやすくすることも目的としているためである．また，往年の 2D ジャンプアクションゲームで実現されていたジャンプの挙動のうち一部は物理エンジンを用いての実装が困難であったことも理由として挙げられる．具体的にはジャンプ上昇時の衝突判定の緩和や，足場をはみ出してのジャンプが該当する．前者については 4.2 節で詳述する．

3.3 可変要素

JumpLab が実行時可変とする 3C の各要素について，以下におおまかな設計指針を示す．具体的な要素については次章で事例を紹介しつつ説明する．

3.3.1 キャラクター

キャラクターはゲームアセットの内でも非常に重要で，特にプレイヤーが操作する「プレイヤーキャラクター」はプレイヤーの感情移入の対象であり，そのデザインが最重要視されている．特に本ツールが対象としている三人称視点の 2D ジャンプアクションゲームにおいてはプレイヤーキャラクターは必然的に画面上にずっと出現し続けており，そのアクションをプレイヤーはゲームを通じて常に目視し続けることになる．加えて，ゲーム中に起きる大半の事象をプレイヤーはプレイヤーキャラクターのアクションまたはリアクションを通じて認識するため [6]，キャラクターデザインは単に見た目の問題ではなく，それを通じてプレイヤーがいかにゲーム世界を感じるかの窓口ともなっている．

しかしながらキャラクターデザインについて，その見た目，例えば形状や色合いなどを含めて可変にしていくと，学習者はその見た目の調整に必要以上に時間をかけてしまう恐れがある．本教材では，学生が省みることが少ないと思われる残り二つの要素について教えることを重視するため，キャラクターデザインについては極力可変要素を少なくすることとした．

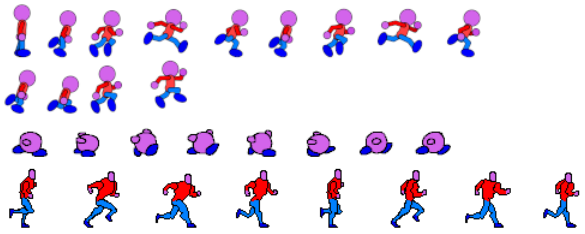

図 3 JumpLab のプリセットキャラクターのアニメーションパターン（部分）．上から 3 頭身・1 頭身・8 頭身キャラクターのもの．

そこで JumpLab では，頭身の異なる 3 種類のプリセットキャラクターを提供するにとどめることとした．**図 3** に各キャラクターのアニメーションパターンを抜粋して示す．異なる頭身のキャラクターを用意した理由は，キャラクターモーションの現実味がどの程度求められるかは，その頭身に強く影響される，つまり頭身の高いキャラクターであれば現実の挙動に近いものが，頭身の低いキャラクターであればそこから離れたものが，それぞれ用いられやすい傾向があると考えたためである．実際にはゲームの舞台設定による影響も強く，例えば宇宙が舞台であれば頭身の高いキャラクターでも地球上のものとは異なるジャンプ軌道が用いられることが多いため，他要素もさらに関係するが，これを深く追求することはやはり 3C の残り二つの要素への集中を阻害するため，本ツールへ盛り込むのは見送ることとした．

3.3.2 カメラ

カメラモーションは，2D 横スクロールジャンプアクションゲームにおいては，キャラクターの動きに追随していかにカメラを並進移動させるか，その動かし方が対象となる．

JumpLab で提供する設定項目は [3] を参考に，10 種の要素を可変とした．ここではその概要を述べる．

図 4 に，JumpLab が採用したカメラモーションのモデルを示す．**Center marker** はカメラの中心位置で，プレイ画面の中心に常に位置することとなる．一般にはプレイヤーキャラクターを追ってカメラは移動することになるが，このときプレイヤーキャラクターと center marker とが重なるように動くのではなく，プレイヤーキャラクターの進行方向に少しずれた箇所を追って動かす場合がある（‘forward focus’）．このとき，カメラ中心を向けたいその箇所をここでは **target focal point** と呼ぶ．また，プレイヤーキャラクターが方向転換をする際，2D ゲームではキャラクターが瞬時にその向きを変えられるものが多く，target focal point の急激な移動を招くこととなり，画面の

[*2] 一部のパラメタ調整操作についてはジョイスティックに割り当てることを可能としているが，ごく限られた要素しか割り当てることができないため本稿での説明は割愛する．

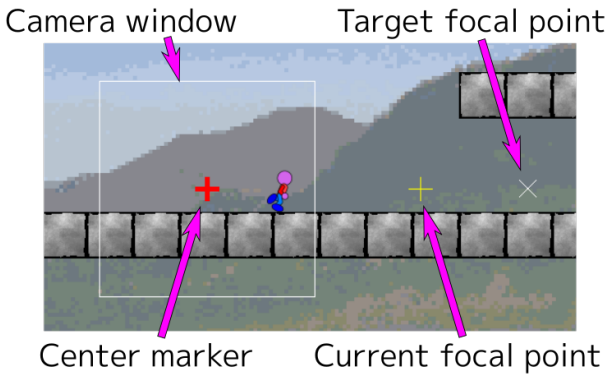

図 4 カメラモーションに関する諸要素を可視化したもの．JumpLab 実行時に学習者はこれを見ながら調整を進めることが可能．

急激な変化を引き起こしかねない．そこで実際のゲームではこれを防ぐために，focal point の移動は滑らかに行われる．このときの中間的な focal point をここでは **current focal point** と呼ぶ．**Camera window** はカメラ中心の周囲に仮想的に設置されたウインドウで，カメラが追う対象（プレイヤーキャラクターや current focal point）がこのウインドウ内にある間はカメラは動かさない．

このモデルの諸パラメタ，例えば target focal point までの距離や current focal point の移動速度，camera window の大きさなど，を JumpLab では可変要素として調整可能としている．これらに加えて，画面全体の変化に大きな影響を与える要素として，キャラクターや障害物が配置された前景部と，背景となる画像が配置される背景部とでスクロール速度を異ならせることで遠近感を演出する ‘parallax scrolling’ についてのパラメタも調整可能とした．parallax scrolling を採用すると背景部の動きは前景部に比べてゆっくりしたものとなるため，前景部の要素の少ないゲームではカメラを急速に動かしてもプレイヤーの視覚的負担が軽減されることが経験的に知られている．

3.3.3 コントロール

JumpLab では 3C の内の「コントロール」について，ジャンプ初速度や重力加速度などジャンプ軌道に関する要素，およびそれらに密接に関連する，プレイヤーキャラクターの移動時における加速度や速度上限などを含めた，20 種の要素を可変とした．

ここで，現実のジャンプでは飛び上がり時の初速度ベクトルおよび重力加速度がその軌道をほぼ決定するが，コンピューターゲームでは遊びやすさや新奇なゲーム感覚の提供などを目的として，単純な放物線軌道に限らない多様なジャンプが用いられている．例えば**図 5** に示すのは，ジャンプによって離れた場所にある足場へと移動することが求められる ‘platform game’ によく見られる軌道で，上昇時にかかる重力加速度が下降時にかかるそれよりも小さい．こうした軌道が用いられる理由は様々に説明されるが典型的には，高い場所にある足場は低いところよりも小さいことが多いため，そこへジャンプで移動する際に微細な操作をするだけの時間的余裕をプレイヤーに与えられること，逆にそこから降りるときは操作難度は逆の理由で低いため素早く降りられた方が余計な時間をプレイヤーに意識させずに済むことが挙げられる．

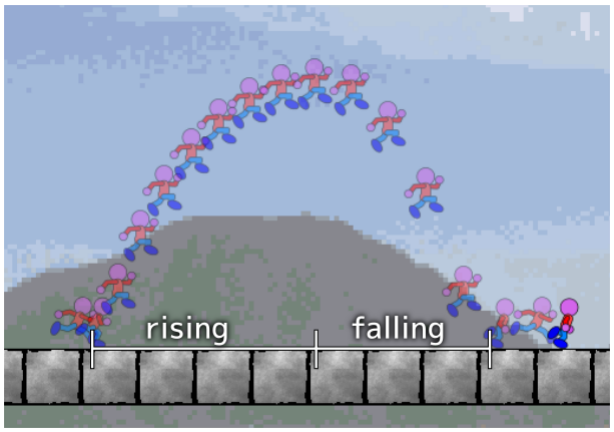

図 5 プラットフォームゲームによく見られるジャンプ軌道．上昇時と下降時とで重力加速度が異っており，上昇にかかる時間よりも下降にかかる時間の方が短い．

また，ジャンプ中の操作がジャンプ軌道に影響を与えるゲームも多数存在するため，そうした軌道を再現するための要素を提供している．例えばジャンプ中の左右移動の可否，またジャンプボタンを離すタイミングによる軌道の変化などがある．

3.4 チャート表示

JumpLab では他に，プレイヤーキャラクターの位置と速度，またカメラの位置と速度を時系列で表示するチャート表示機能を提供している（**図 6**）．3C のうちカメラとコントロールについて，プレイヤーによる操作の結果がどのように反映されるのか，その時系列変化を 280 フレーム（4.67 秒）ぶん表示する．プレイヤーキャラクターやカメラの動きの変化の滑らかさ，またキャラクターの移動にどの程度の時間遅れをもってカメラ移動が追随しているのかをよりわかりやすく確認することができる．

3.5 設定保存およびプリセット機能

学習者が調整した可変要素には，名前をつけて保存することができ，また後でそれを読み出すことが可能である．またこの機能を利用して，JumpLab ではあらかじめ作成された，既存の著名ゲームのジャンプ軌道を模倣したプリセットデータが提供されている．このプリセットデータは，文献 [5] に掲載された，既存ゲームのジャンプ軌道のグラフを参照し，また実際のゲームのプレイ画面を録画し，コマ送りでジャンプ軌道を確かめながら著者らが手作業で作成した．

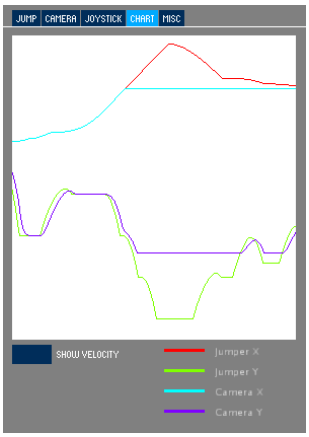

図 6 チャート表示機能．ここではプレイヤーキャラクターとカメラのそれぞれの X 座標・Y 座標が表示されている．トグルスイッチ操作によって，それぞれの速度の X 成分・Y 成分の表示に切り換えられる．

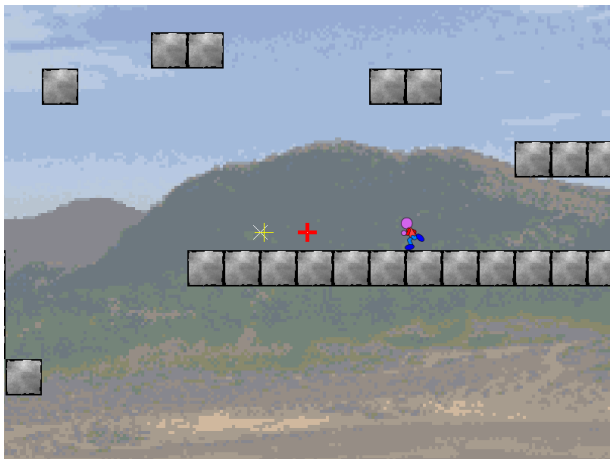

図 7 Forward focusing の例．プレイヤーキャラクターから進行方向へ 200 ピクセル離れた箇所に target focal point が配置されている．前方視界を確保できるカメラモーション．

4. JumpLab で作成できるジャンプアクション・カメラモーションの例

本章では JumpLab が提供する可変要素を調整することで得られる様々な 3C について例示し，その過程で一部の可変要素について詳述する．本論文で詳述されていない要素については，JumpLab 配布サイトで簡単な説明をしているのでそちらを参照されたい．

4.1 Forward focusing

Forward focusing は多くの 2D スクロールアクションゲームで採用されているカメラモーションの手法である．同種のゲームにおいては進行方向から障害物や敵キャラクターがスクロールに応じて出現することが多い．このとき，突如出現したそれらオブジェクトとプレイヤーキャラクターとの距離が短か過ぎると，それらへの対処を行うだけの時間的余裕がプレイヤーに与えられず，プレイヤーに不満をもたらす．そこでプレイヤーキャラクターの前方視界を広くとるためにカメラ中心をプレイヤーキャラクターから進行方向にずらすのが forward focusing である．JumpLab では target focal point までの距離を ‘focus distance’ 変数で調整できる．

Forward focusing を採用した場合に問題となるのは，プレイヤーキャラクターが急激に進行方向を変えた場合に target focal point が急速に位置を変えることになるため，カメラ位置が急速に変化することにある．カメラ位置の急激な変化はプレイヤーにとって不快な体験の原因となりうる．特に頻繁な方向転換によって画面が振動する場合は，いわゆる「映像酔い」を引き起こすことが知られるため [7]，極力これを抑える必要がある．

急激なカメラ位置変化を抑える方法として単純な手法は，カメラ移動を easing させるもので，JumpLab では easing の強さが可変となっているのでこれは実現が容易である．しかし単純に easing をかけた場合，カメラ中心は target focal point からプレイヤーキャラクターの進行方向の反対側にずれることとなり，easing の強さによってはカメラ中心がプレイヤーキャラクターよりも遅れてしまうことになり，forward focusing を達成できない．そこで，図 4 で示した current focal point を，target focal point を追いかけるように滑らかに変化させることで，forward focusing を維持することができる．JumpLab では current focal point の移動速度を可変にすることでこの効果の強弱を調整可能とした．

Forward focusing を用いて，前方視界を十分に確保するためには，プレイヤーキャラクターの水平方向の移動速度や加速度も深く関係する．ゆっくりと歩いての移動が中心となるゲームであれば，前方視界を広くとる必要性は低く，速く走って移動するゲームであれば前方視界は広くとりたい．また，前方から出現した障害物をジャンプで避ける場面であれば，ジャンプ力と回避しやすさが関係するため，やはり前方視界の調整とジャンプ力とが関係することとなる．これらのことから，3C の各要素は相互に関係しあい，それらを同時に調整する必要があることがわかる．

4.2 ジャンプ上昇時の衝突判定の緩和

プラットフォームゲームのうち，『スーパーマリオブラザーズ』に代表されるように，高い場所に配置された足場の上にジャンプで移動することが求められるようなゲームにおいて，**図 8** に示すようにブロックの下から垂直ジャンプで移動したくなることがある．横から助走をつけて斜めにジャンプするのが基本だとしても，左右の障害物が邪魔になっていたり，走りながらのジャンプ操作に不安のある場合などで，こうした要求が生じやすい．

このとき，プレイヤーキャラクターとブロックとの衝突判定において両者が衝突した瞬間にジャンプの上昇を停

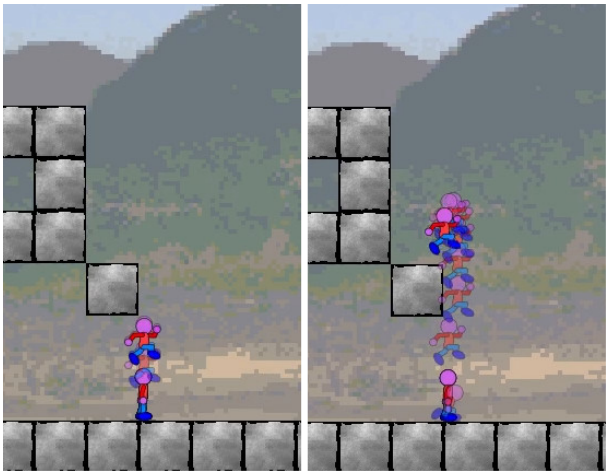

図 8 衝突判定の緩和の様子．左図では緩和をしておらず，ジャンプしてプレイヤーキャラクターの頭が頭上のブロックに衝突した際にジャンプが停止し，直後に下降を開始している．右図では頭上のブロックに衝突後，ブロックを避けるようにプレイヤーキャラクターが右に並進移動させられた後，上昇が継続している．その後プレイヤーによる空中操作でキャラクターが左に移動し，頭上にあったブロックの上に着地しようとしている．

止し下降させ始めるようなアルゴリズム（図 8 左）だと，ジャンプ前にプレイヤーキャラクターの位置を厳密に調整する必要がある．

ここで，衝突時にプレイヤーキャラクターの位置を水平方向に補正してブロックを回避し，かつ上昇動作を継続させるようにすると（図 8 右），プレイヤーキャラクター位置の微調整が不要であり，またその後にブロック上へと移動するのが容易となる．このような衝突判定の緩和は多くのジャンプゲームで採用されている．

この緩和処置は，ジャンプ中に空中でのプレイヤーキャラクターの水平位置の微調整を許す操作とセットで行う必要がある．空中での移動操作はあまり自由に行えるようだと現実でのジャンプ軌道から大きく逸れたものとなるため，プレイヤーキャラクターの頭身の高い，現実の挙動に近い動きが求められるようなキャラクターの場合にはあまり馴染まない恐れがある．一方で，頭身の高いキャラクターの形状は，一般に衝突判定用に用いられる矩形のバウンディングボックスと大きく異なるため，この緩和処置がないと見た目にはぶつかっていない障害物にぶつかってジャンプが中断されたように見えてしまう場合もある．これもまた 3C の各要素が相互に関係しあう事例の一つである．

5. Unity 版の開発

JumpLab は Processing で開発したものだが，ゲーム開発を教える現場では種々のゲーム開発専用のエンジンが広く用いられており，またそれらは実際のゲーム開発の現場でも用いられているため，そうしたゲームエンジン上で動作する教材の需要の高いことが予想される．

もっともそれらゲームエンジンでは，JumpLab が提供

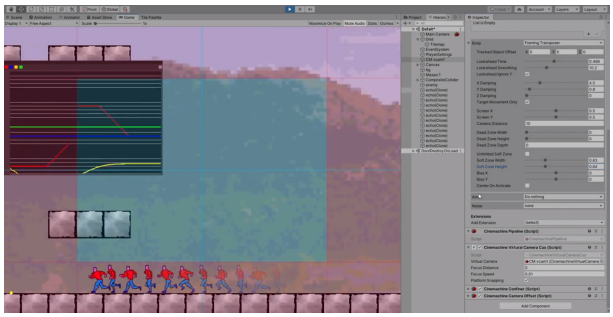

図 9 Unity 版の JumpLab の開発中の画面．

しているようなプレイ中のパラメタ調整機能を標準で搭載しており，開発に慣れた者であれば特に専用のツールがなくとも 3C の調整に取り組むことは容易い．

一方で，初学者にとってはこうしたゲームエンジンはできることが最初から多様で，3C に集中した試行を学ぶには阻害要因が多いとも考えられる．実際，著者らが所属する学部ではゲームエンジンの Unity を用いてゲーム開発に取り組む学生が少なくないが，3C の調整以前に，Unity を用いてのプログラミングによるゲーム開発の学習に時間を多く投資し，調整の時間を確保しない，あるいはできない傾向が見受けられる．

そこで現在，JumpLab を Unity で再実装し，Unity の長所を活かしつつ，3C の学習に集中できるようなツールの開発に取り組んでいる．ここではそれについて簡単に報告する．

図 9 に開発中の Unity 版 JumpLab の画面を示す．画面構成は Unity で一般的な開発用画面に即したものとなっている．パラメタの操作については，Processing 版では独自に GUI を構築して提要したが，Unity においてはゲーム実行中のパラメタ操作で標準的に用いられる inspector を用いて実装しているため，Unity での開発経験があればすぐに理解できるようになっており，また独自にパラメタを追加した際にもその GUI を inspector に追加することが容易となる．また，Processing 版にはレベルエディタが付属しておらず，ステージの形状は起動時に CSV ファイルから読み込まれた後固定となっていたが，Unity 版ではやはり標準で提供されているレベルエディタを用いて編集することができるため，例えばジャンプ力を調整した後にそのジャンプ力に見合うようにステージ形状を改善する，といった実践的な演習が可能となる．

一方，Processing 版では独自に開発していたジャンプ処理や衝突処理は，現在こうしたゲームエンジンを用いて開発されるジャンプアクションゲームのそれとは大きく異っているため，Unity 版にこれを忠実に移植するのは馴染まないと判断し，全面的に作り換えることとした．現在多くのゲームエンジンではジャンプアクションゲームを物理エンジンを用いて実装するのが一般化している．そのため，物理的な挙動をそのまま再現する上では物理エンジンの機

能をそのまま用いることができるため実装が単純化される．一方で，現実の物理に反する挙動，例えば衝突判定の緩和処置は Processing 版の処理をそのままでは再現できない．こうした点については Processing 版の忠実な移植にはこだわらず，現在主流の手法に即した形に随時変更しながら改訂を続けている．

6. 議論

現時点で，JumpLab はあくまでも 3C を教えるための補助的な教材でしかない．本ツールを用いて具体的にどのようにして 3C の重要性を教えるか，体系的な教材の開発までには至っていない．著者らはこれまで大学での講義で JumpLab を用いてパラメタ調整の重要さを教えているが，使用を開始したのが 2020 年 7 月であり，同講義は新型コロナウイルス感染症 (COVID-19) 対策のためビデオ講義となっており，実際に受講者がどのように JumpLab を操作して学習を進めたかについての観察ができておらず，明示的なフィードバックも得られていない．これらについては今後の課題とする．

3C の考え方は 2D ジャンプアクションゲームに限定されるものではない．多くの学生は 3D ゲームの開発に強い興味を持っているため，3D ゲームを対象とした教材が必要である．3D ゲームにおけるカメラモーションについては，ゲーム空間内のオブジェクトとの干渉が多く，調整すべきパラメタが多岐に渡ってしまうため，3C を教える教材としてどのような要素を重視すべきかについては議論が必要となろう．

本ツールは 3C に基いたゲームデザインのための教材として開発が開始されたが，公開後に寄せられた意見として，ゲーム感覚を認知科学や視覚心理学の観点から科学的に検証するためのツールとして使えるのではないかというものがあった．著者らもその応用には強く興味を持っており，今後の課題としたい．なお，現在の実装が抱える問題点としては，フレームレートが安定しない場合があること，展開が単調で被験者がすぐに操作に飽きてしまいかねないことが挙げられる．これらについて，現在開発を進めている Unity 版では部分的な解消を試みている．後者については，レベルを広くした上で簡易な敵キャラクターの追加を試験している．またゲームエンジンをベースとしているため，基本的なゲーム要素の追加は容易い．

7. まとめ

本論文では，コンピューターゲーム開発の初期段階で重視される 3C (Character, Camera, Control) の重要性について教えるためのインタラクティブ教材として，2D 横スクロールジャンプアクションゲームを題材とした「JumpLab」について，その設計指針や細部を報告した．

今後は本ツールを用いての教材開発を進め，その効果についての検証を計画している．また実践的なゲーム開発に用いられる開発環境での学習を求める者のために，現在主流のゲームエンジン向けの再実装を進めているが，この完成と公開を短期的な目標としている．

参考文献

[1] Cerny, M.: Method, *D.I.C.E Summit 2002*, Academy of Interactive Arts & Sciences, (online), available from ⟨https://www.youtube.com/watch?v=QOAW9ioWAvE⟩ (2002).

[2] Fasterholdt, M., Pichlmair, M. and Holmgøard, C.: You Say Jump, I Say How High? Operationalising the Game Feel of Jumping, *Proc. of the First International Joint Conference of DiGRA and FDG*, (online), available from ⟨http://www.digra.org/wp-content/uploads/digital-library/paper_248.pdf⟩ (2016).

[3] Karen, I.: Scroll Back: The Theory and Practice of Cameras in Side-Scrollers, *GDC '15*, GDC, (online), available from ⟨https://www.gdcvault.com/play/1022243/Scroll-Back-The-Theory-and⟩ (2015).

[4] McEntee, C.: Rayman Origins, *Game Developer Oct. 2012*, pp. 26–31 (2012).

[5] Summerville, A., Osborn, J., Holmgård, C. and Zhang, D. W.: Mechanics Automatically Recognized via Interactive Observation: Jumping, *Proc. of FDG '17*, New York, NY, USA, ACM, (online), DOI: 10.1145/3102071.3102104 (2017).

[6] Swink, S.: *Game Feel: A Game Designer's Guide to Virtual Sensation*, Morgan Kaufmann (2008).

[7] 電子情報技術産業協会：映像酔いガイドライン検証システムの開発に関するフィージビリティスタディ報告書，財団法人機械システム振興協会 (2007).