Unity®

ヘブンバーンズレッド
導入事例

Luaと自作ビューアで作り込んだ『ヘブンバーンズレッド』の裏側——Unityの拡張性を活かすWright Flyer Studiosの開発メソッド

2022年2月10日、『消滅都市』や『アナザーエデン　時を超える猫』（以下、『アナザーエデン』）など、数々のヒットタイトルを世に送り出してきたWright Flyer Studios（株式会社WFS）の新作『ヘブンバーンズレッド』がリリースされました。同タイトルはアニメ化もされたゲーム『CLANNAD』などを手掛けたシナリオライター・麻枝 准氏による「15年ぶりの完全新作」として注目を集め、リリースから3日で早くも100万ダウンロードを達成。現在も多くのユーザーを楽しませています。

本記事では、Wright Flyer Studiosの『ヘブンバーンズレッド』開発チームにインタビュー。お話を伺ったのは、テクニカルディレクターを務める西田綾佑氏、メインプログラマの奥村典史氏、フィールド・グラフィックスプログラマである佐藤真也氏です。
開発のいきさつから、ストーリーやフィールド設計におけるこだわりなど、制作の裏側を伺う中で見えてきたのは、Unity上でLuaや自作ビューアなどを活用した、このゲームならではの開発メソッドでした。

エンジニア全員でディスカッションしてUnityを採用

『ヘブンバーンズレッド』の開発は、いつ、どのようにして始まったのでしょうか。

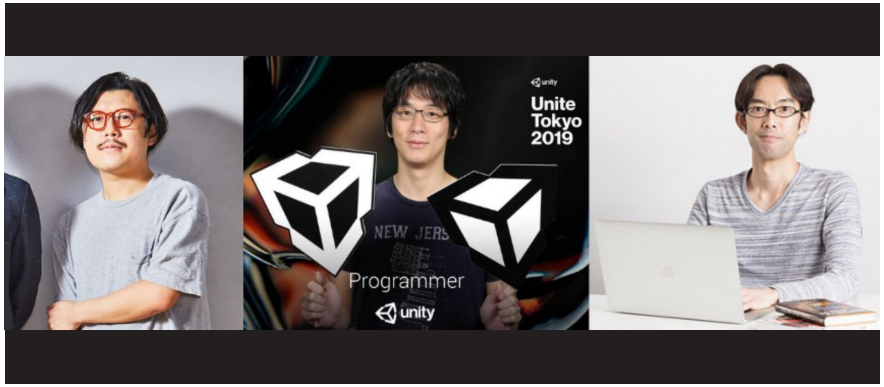
西田：元々は、約5年ほど前にプロジェクトが立ち上げられました。その後Wright Flyer Studiosで開発を担当することに決まったのが3年半くらい前のことですよね？

奥村：そうですね。WFS内でプロジェクトが立ち上がったのはそれくらいの時期で、プロデューサーともう1名体制で始まりました。そこにディレクターと、アートディレクター、プログラマーである私が加わって、しばらくの間は計5名でプロジェクトを推進していました。

WFSが開発を担当するようになった時点では、どれくらいの進捗だったのでしょう？

奥村：開発としてはゼロからのスタートです。シナリオプロットはあったのですが、それ以外は何も決まっておらず「どのような環境で開発を進めるか」から考えなければなりませんでした。議論を進める中で『アナザーエデン』で培ったJRPGの環境ベースでの開発を検討したのですが、WFSでメンテナンスしていた既存のゲーム制作フレームワークでは表現の限界を感じていました。

西田：当時、Wright Flyer Studiosでは『ヘブンバーンズレッド』を含め、3本のゲーム制作が走っていました。それぞれの開発環境を決定するために、エンジニア全員でディスカッションをした結果、『ヘブンバーンズレッド』ではUnityを採用することに決めたのです。



（Wright Flyer Studios テクニカルディレクター 西田綾佑氏、メインプログラマ 奥村典史氏、フィールド・グラフィックスプログラマ 佐藤真也氏）

Unity導入の決め手はエディターとチームの“拡張性”

エンジニアのみなさんでUnityを採用することに決めたのですね。決め手は何だったのでしょうか。

佐藤：私は『ヘブンバーンズレッド』のプロジェクトに加わる直前まで、育休を取っていて、その期間にUnityを触っていたんです。その中で感じた魅力が2つあります。まずは、豊富なアセットが用意されている点。そして、CRIWAREが安定して動作する点です。それに、インターネット上でたくさんの導入事例が紹介されていて、情報が入手しやすかったこともUnityを推す理由の一つになりましたね。よほど特殊なことをしない限り、導入は妥当だと感じていたので、私も賛成しました。

奥村：『ヘブンバーンズレッド』に携わる前からUnityを使っていて、拡張性の高さが最大のメリットだと感じていました。
具体的には、容易にさまざまなツールと組み合わせることができること。C#でつくったツールをそのままUnityのエディター上で動かせるのですが、他のゲームエンジンではそうはいきません。

西田：あとは、Unityを使えるエンジニアが多いことも導入のメリットですよね。ゲームがヒットして、チーム体制を強化しようとなったとき、自社エンジンを使っているとなかなか人数を増やせません。一方、Unityであれば、多くのエンジニアが経験を持っているため、増員しやすい。中長期的な目線でも、Unityを導入することが最適だと考えました。

今だから振り返れる苦労はありますか？

西田：Unityの活用という点では苦労はあまりなかったのですが、開発の初期段階では壁に直面していましたね。WrightFlyer Studiosの開発ステップは、「α開発」と呼ばれる工程から始まります。まず、少人数体制でモックを制作し、上層部がモックを見て「β開発」、つまりは本格的な開発に進む判断を下しているのですが、『ヘブンバーンズレッド』はなかなかβ開発に進めない時期がありました。

奥村：「α4」までいきました。期間にすると1年半ほどの間は、やり直しの日々だったわけです。この間は、2Dゲームとして制作が進められていました。

その後、チームに開発統括を務める下田翔大が加わってから「このシナリオは3Dグラフィックスで表現されるべきだ」「フィールド上でもシナリオを表現するためにプランナーがLuaを組んで開発するべきだ」と判断し、改めて開発を進めたところ、β開発へのGOサインが出て、そこから一気に前進していきました。

少ないデータ量で、自由度の高い進行を支える工夫

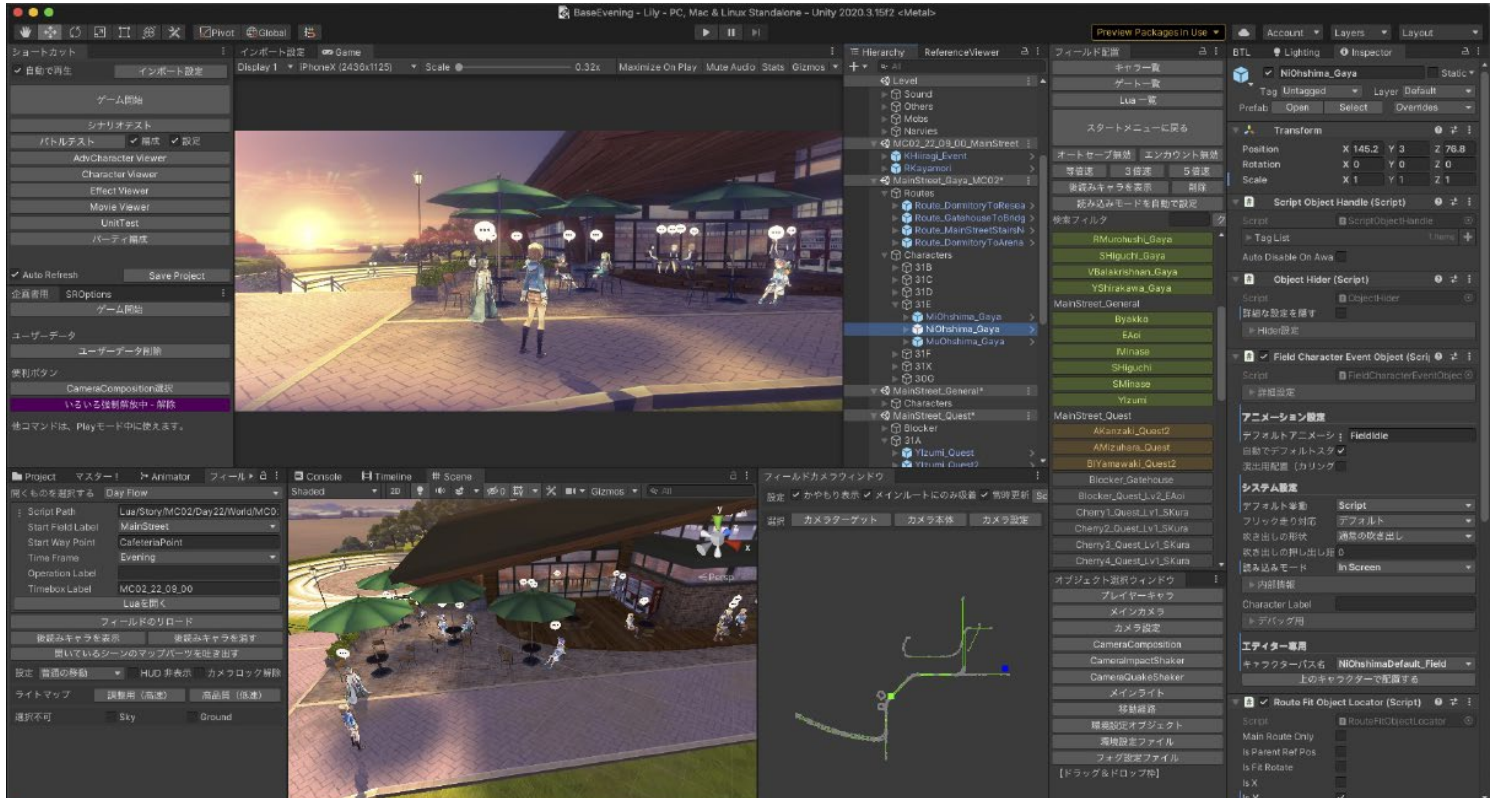
開発の下地が整ったわけですね。そうして開発された『ヘブンバーンズレッド』の特徴は、どのような点にあるのでしょう？

佐藤：まず、ストーリーパートのフィールドに関しては、キャラクター配置のパターンが多いことが挙げられます。このゲームは、広いマップを旅する一般的なRPGとは違い、基本的には「主人公たちが所属する部隊の基地」という単一の場所を中心に物語が進んでいきます。

また、空間の移動に即してシナリオが進むのではなく、ゲーム内の時間経過と共にシナリオが進んでいくシステムです。ゲームは1日単位で進み、「早朝」「午後」「夕方」といった13区分で条件別にイベントが発生します。

特徴的なのは、過去に戻れること。（取材時の2022年2月下旬）現在、第3章までプレイできるようになっているのですが、その章を進めている最中だとしても、いつでもホーム画面から第1章のDAY1にも、第2章のDAY10にも戻れるようになっているのです。

フィールド編集の画面



プレイヤーからすると、イベントの取りこぼしがなくなる一方、自由度は高いですね。

佐藤：複雑な構造になっています。しかも、主人公たちがいる基地には多くのキャラクターを配置しているので、「どの時間」「どの場所に」「どのキャラクターを」置くかによって、途方も無い数のパターンが存在するんです。

なるべく少ないデータ量でこの仕組みを実現するために、Unityの複数シーンを読み込み・編集する機能を駆使し、多くて7つのシーンを同時に読み込むようなシステムになっています。

また、それらのシーンをすべて手作業で重ねていくのは不可能に近いので、「フィールド編集ウィンドウ」というツールを自作。これによって、編集したいフィールドとストーリーを選択すれば、あとは自動でパターンを生成できる仕組みを実現できました。

フィールド内にかなりの数のキャラクターを配置しているので、メモリ負荷を減らすべく、キャラクターが画面に入りそうになったら読み込み、画面から離れた瞬間に読み捨てるといった、メモリ負荷を下げる処理をしています。大量のキャラクターを配置し、基地内のにぎやかな雰囲気を伝えられるようになりました。

Luaの活用によって、複雑なゲームシステムを実現

RPGは街から街へ移動することで、ストーリーを進めるものが多いイメージがありますし、いつでも自由に過去の特定の時点に戻れるシステムも、あまり聞いたことがないですね。

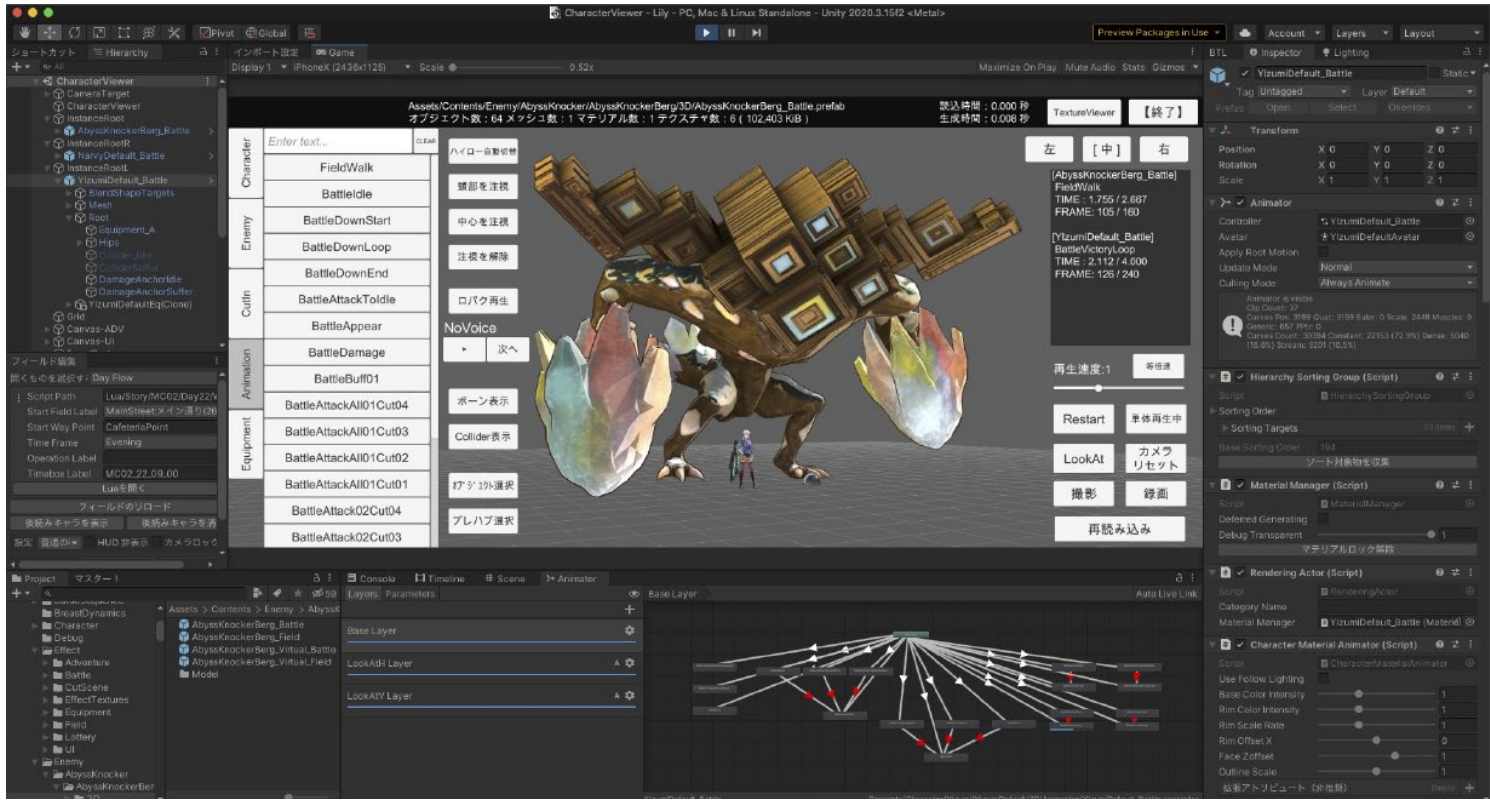
佐藤：そうなんですよ。最初にこのシステムを聞いたときは「本気か！？」と思いました（笑）。制御が大変になることは明白だったので。

奥村：いわゆるRPGのストーリーが横に広がっていくイメージだとすれば、『ヘブンバーンズレッド』のストーリーは縦に深まっていくイメージ。RPGのようでいて、その実、ノベルゲームだと言えると思います。

当然、基地フィールドのキャラクター配置だけではなく、キャラクターたちの発言に関わるノベルパートの分岐も制御しなくてはなりません。それらの途方も無い分岐を制御するためにもLuaが役立ちました。

『ヘブンバーンズレッド』では、Luaをフィールドや演出の制御だけではなく、ノベルパートの分岐をコントロールするためにも使用しています。Luaで「ストーリーがどこまで進んだら」「どんなイベントを呼び出すか」を書いているわけですね。

キャラクタービューアの画面



西田：Wright Flyer Studiosでは他のタイトルでもLuaを積極的に活用しているので、スタジオ全体の特徴と言えるかもしれません。

奥村：Luaの書き方を定めることが、今回の開発における山場のひとつでした。最終的には、スクリプターがLuaを組めるようにしなければならないので、プランナーからエンジニアも交え、ディスカッションを繰り返しましたね。

イテレーションを早く回すための「自作ビューア」

複雑性の高いシステムで、みなさんにとっても新たなチャレンジだったと思います。イテレーションを早く回し続ける必要があったのではないでしょうか？

佐藤：そうですね。重要な役割を果たしたのが、Unity上の自作ビューアです。『アナザーエデン』でも、イテレーションを早く回すために自作ビューアを導入していて、今回はキャラクターのビューア、戦闘シーンのエフェクトのビューアなどを制作しました。

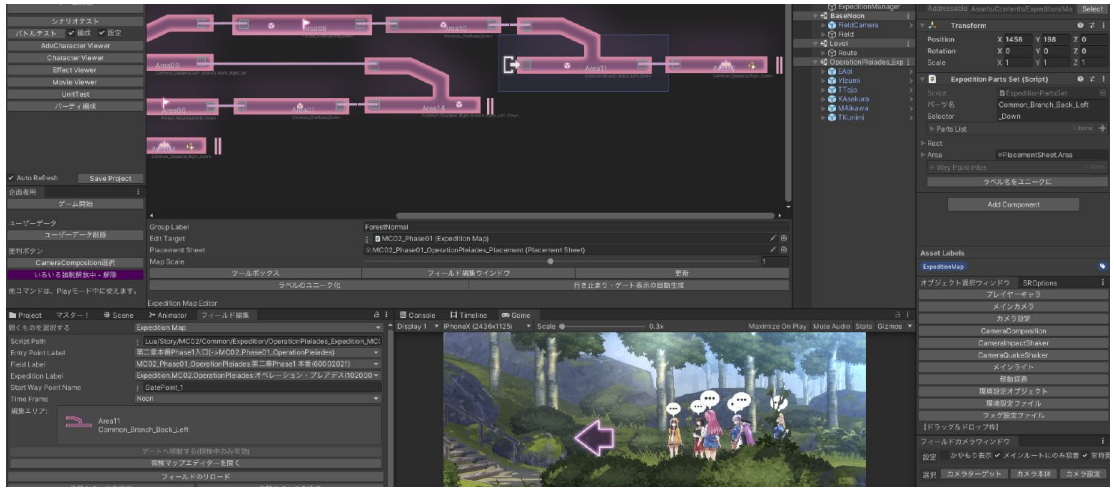
なぜ、自作ビューアがイテレーションを早く回すことに寄与するのでしょう?

佐藤：ビューアがないと、アーティストがキャラクターをつくったあと、ゲーム画面上で造形を確認するためにはプランナーにデータを渡し、ゲームに組み込んでもらわなければならず、かなりの時間を要してしまいます。ビューアをつくることによって、その工程を省くことができるわけです。

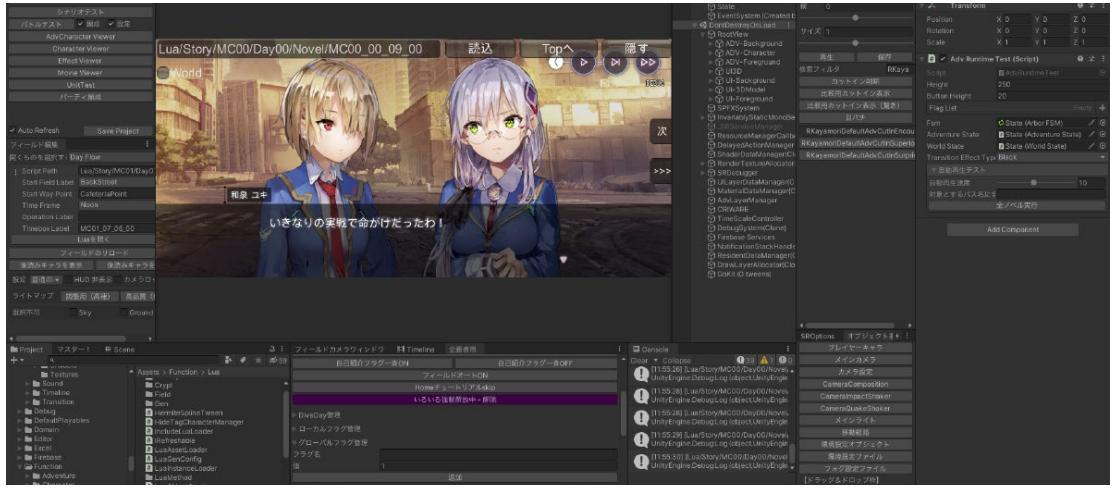
キャラクターの基本造形だけではなく、武器を装着した様子や首を動かす動作などを確認できる体制を整えて、開発を進めました。

奥村：その他にも、ダンジョンを探検する際に表示されるマップを制作し、確認するためのビューアもつくりました。Unity上のマスターデータからさまざま情報を取り出し、それらを組み合わせることでマップをつくれるような機能を実装しています。

奥村：あとは、ノベルビューアも重要な働きをしてくれました。先ほど、ノベルパートの分岐はLuaによって制御しているというお話をしましたが、それらのLuaを単体ごとに確認できる仕組みを整えたことによって、イテレーションを早く回せたと思っています。



探検マップビューアの画面



ノベルビューアの画面

西田：イテレーションを早く回す工夫としては、ホットリロードの実装も挙げられます。Unity　Editorを再生したままLuaを編集して、ホットリロードを実行すると編集されたLuaを読み込んで再生されるような仕組みを整えて開発を進めていました。

つまり、エディターは再生したまま、データを編集できるようになっていて、これはかなり特徴的な点かもしれません。

奥村：そういった仕組みを実現できる懐の広さも、Unityの良さですよね。

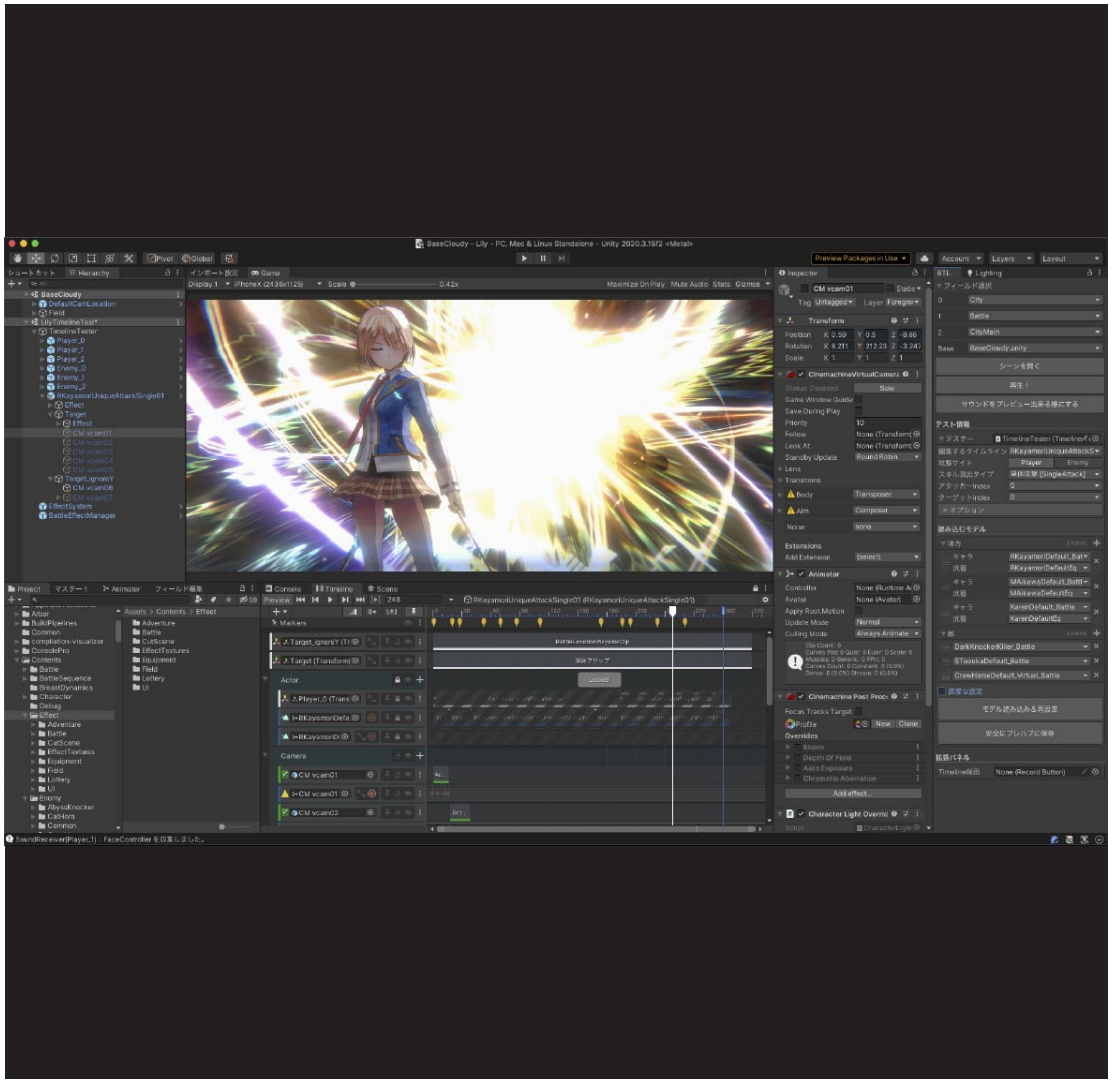
誰でも簡単に機能拡張ができる

先ほども「Unityを導入するメリットは拡張性の高さにある」というお話がありました。具体的に、その拡張性が開発に活かされた例があれば教えてください。

佐藤：自作ツールやビューアがとてもつくりやすかったですね。工数をかけずにサクッとつくることができましたし、インターフェースが洗練されているので、とても使いやすいものになりました。

奥村：プランナーがデバッグ機能を勝手に自作し始めたとき、その拡張性の高さを実感しました。自分用のエディター拡張ウィンドウをつくり出したんですよ（笑）。最終的には開発に関わるほとんど全員が、そのプランナーがつくったデバッグ機能を利用していました。

佐藤：Unity導入前は自作のアプリケーションという形でツールを開発していたのですが、それがかなり大変だし、インターフェースもちぐはぐで決して使いやすいとは言えなかった。Unityを導入したことで、簡単に使いやすいビューアがつくれるようになったので、そこはとても助かりましたね。



Timelineにおけるスキル演出編集の画面

西田：あとは、Timelineが優秀ですよね。いわゆる必殺技などの見せ場シーンの演出は、基本的にTimelineで制御しています。「どのタイミングで」「どういったエフェクトを出す」といったことをコントロールしているのですが、これを自作ツールでやろうとすると、とても大変なんですよ。

そういった自分たちでは簡単につくれない機能を、デフォルトの状態で備えているこ

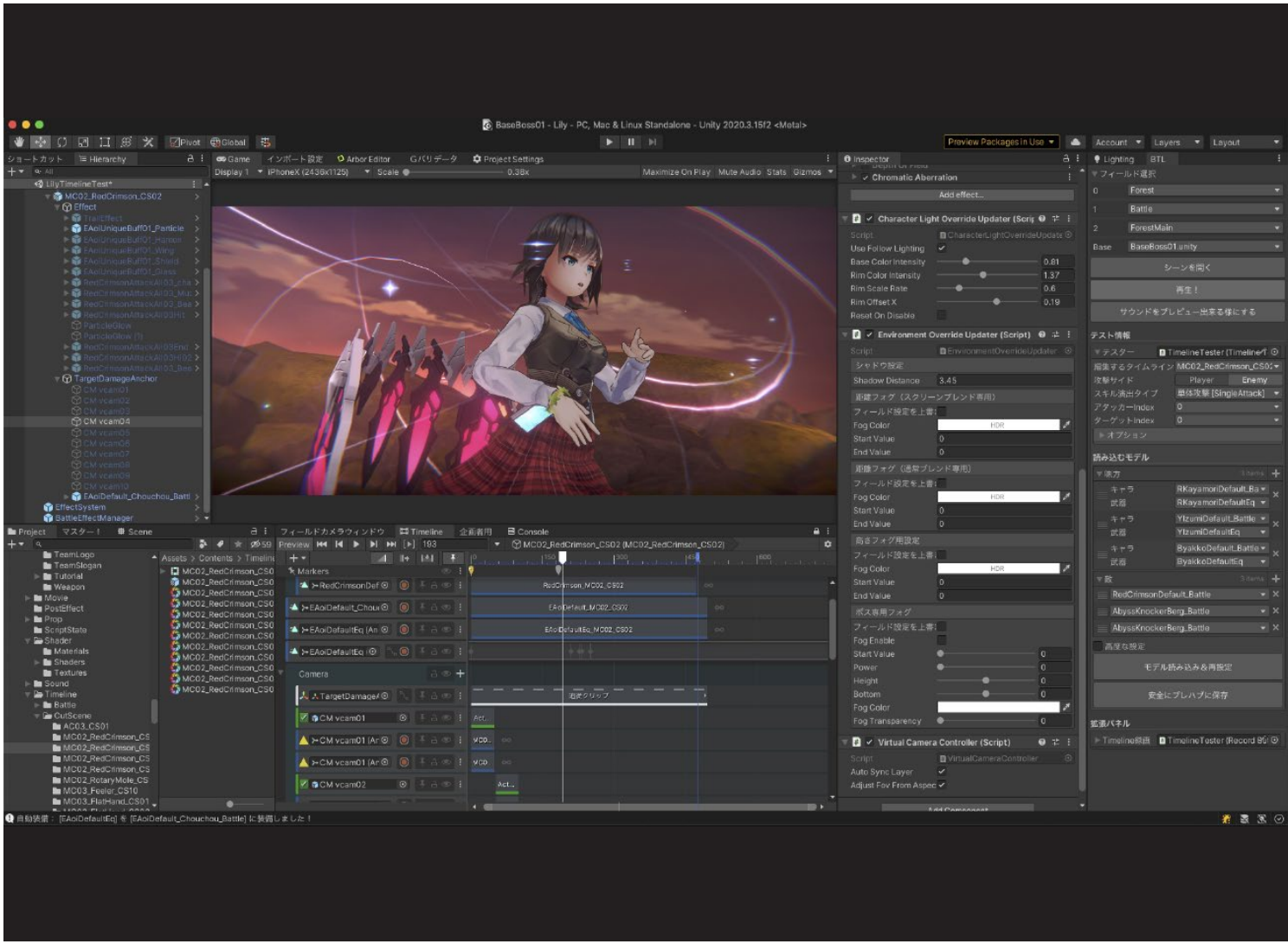
とがありがたいですし、足りなければ自分たちの手で簡単に拡張できる。ユーザーに最も見せたいシーンのクオリティを上げるための強力な武器になり、他のゲーム開発でも大いに活用させてもらっています。

Unityをかなり効果的に活用していただいていて、私たちとしてもとても嬉しいです！　最後に、Unityへの要望があれば教えてください。

奥村：使い始めた当初からとても使いやすいゲームエンジンだという印象がありましたが、現在はさらに成熟度が増しましたよね。

ゲーム開発に必要な機能は一通り揃っていますし、既存機能をさらに強化したり、機能同士の連携を高めたりする方向に進化してくれたら嬉しいなと思っています。今後もUnityを活用して、さらにユーザーが楽しめるゲームを生み出していきたいですね。

（画像提供：Wright Flyer Studios）



Timelineにおけるカットシーン制作の画面