新しい 3 次元アプリケーションをつくるための CosmoAPI の設計とその評価

Design and implementation of CosmoAPI for making new 3D applications

薬師寺浩二 † 前田良史 † 南迫博和 † 小出洋 ††

Koji YAKUSHIJI† Yoshifumi MAEDA† Hirokazu MINAMISAKO† Hiroshi KOIDE††

† 九州工業大学大学院情報工学研究科情報科学専攻

†† 九州工業大学情報工学部知能情報工学科

†Dept. of Artificial Intelligence, Graduate School of Computer Science Systems Engineering, Kyusyu Institute of Technology

††Dept. of Artificial Intelligence, Faculty of Computer Science and Systems Engineering, Kyusyu Institute of Technology

k_yakushiji@mickey.ai.kyutech.ac.jp

我々は LG3D 上で斬新な 3D アプリケーションを開発するために，3D アプリケーションを作るための便利な API（CosmoAPI）を設計した．CosmoAPI は開発者が思い描く独創的なインターフェースを容易に実現することを目的として設計している．我々は CosmoAPI を用いて「Cosmo Scheduler D」を開発し，JavaOne にて高い評価を得た．本論文では，この Cosmo Scheduler D を紹介し，CosmoAPI による GUI の表現が LG3D の API を直接使うよりも有効であることを示す．

1 はじめに

私達の目的は，有用な 3D アプリケーションを開発する際に必要となるであろう機能を API として用意し，それを使ったアプリケーションを開発し評価することである．その結果，同じプラットホーム上で同時期に開発されている他のアプリケーションよりも完成度などで評価されるアプリケーションを開発することができた．

近年の CPU とグラフィックス性能の向上は著しいものがあるが，それにも関わらず現在のデスクトップ環境の概念は 20 年前もの昔から変化していない．私達はコンピュータの性能の向上によって，新たな GUI 環境が実現できるだろうと考えた．3D デスクトップは昔から考案されてきたが，マシン負荷や画期的なアイデアの不足の為あまり重視されず，実用化されることは少なかった．しかし近年になって LG3D（2.2 節 [1] ），XOrg6.8[2] など，Sun Microsystems を始め大企業も新しい GUI に注目するようになってきた．

LG3D はオープンソースプロジェクトであり，アプリケーション開発者とも連携を取りながら活発に開発が進められている．また LG3D は Java3D（2.1 節 [3]）をベースにしており，私達も以前から Java3D を使って 3D アプリケーションの研究をしてきている．よって私達は 3D アプリケーションを開発するにあたり，最も理想に近い 3D デスクトップ環境である LG3D をプラットホームとして選択した．

3D デスクトップを生かすには，3D の特性を活かしたアプリケーションが必要不可欠である．しかしそのような 3D アプリケーションを作る場合，3D デスクトップに依存した機能をいちいち調べる必要があり，大きなコストがかかる．実際 LG3D 上で開発するにあたり，以下の機能を実装するための十分な API が用意されていないことがわかった．

A．3D 空間における視点，オブジェクトの操作
ユーザが画面上の三次元空間を容易に認識でき，二次元のデスクトップよりも多くの情報を分かり易く豊かに表現する事が理想的である．二次元のモニタ画面に表示されているにも関わらず三次元らしい動きを表現することが必要で，これによりユーザが三次元空間を容易に認識できるようにする．

B．ジェスチャによる入力支援
3D デスクトップでは現在最も普及している入力デバイスであるマウスを主に使用する．二次元的な入力デバイスであるが，三次元空間におい

てシームレスな操作感を得るためにアプリケーション単位でのジェスチャによる入力は必須である．

また，Java3D + LG3D によって改善された 3D ソフトウェア作成上のサポート機能を以下に挙げる．

Java3D

1. プラットホームに依存しない 3D グラフィックスを実現
2. OpenGL 等のプリミティブなグラフィックスライブラリを直接扱う必要が無い設計
3. 三次元空間の描画に必要な情報を一つの木（シーングラフ）として扱うことが可能
4. 3DCG の世界で主要なモデルデータフォーマットに対応したローダを用意

LG3D

1. クライアント/サーバシーングラフを用意することによる，複数アプリケーションでの仮想三次元空間の共有を実現
2. アニメーションを容易に実現するためのプリミティブを用意
3. 三次元でのレイアウト管理を用意

しかしながら，前述の A と B を実装するための API は今のところ用意されていない．そこで私達はこれらを実現するための API モジュールを用意した．

A．3D 空間における視点，オブジェクトの操作
関数として定義した三次元空間上の直線や曲線に沿ってカメラの移動やオブジェクトの配置をする．これによってプログラマは思い通りのカメラワークやアニメーションを実現できる．

B．ジェスチャによる入力支援
LG3D ではアプリケーションを操作するためのジェスチャ機能を備えているが，それはシーンマネージャのための機能で，一般にアプリケーションの中からは利用できない．そこでジェスチャ機能をアプリケーション内部から利用できるようにした．

これら API の効果を評価するために，この API モジュールを活用した 3D アプリケーション「Cosmo Scheduler D」を作成した（図 1 ）．Cosmo Scheduler D は 3D デスクトップ空間を宇宙空間とし，そこに浮かぶ太陽系をスケジュール帳に見立てたスケジュール管理ソフトウェアである．ボタンやメニューを極力廃したシンプルなインターフェースを備え，3D デスクトップならではの奥行きのある表現や操作感を有している．現在発表されている他の LG3D 上のアプリケーションと比較して評価を行った結果，本 API による表現の効果は大きく，有効であることがわかった．

図 1: Cosmo Scheduler D

2 関連研究

2.1 Java3D

Java3D は Sun Microsystems が提供する Java 言語向けの三次元グラフィックスの拡張 API である．Java であるので特定のプラットホームに依存せずに 3D グラフィックスを扱うことができる．レンダリングの処理はリアルタイムで実行され，実際の描画は OpenGL や DirectX などの 3D グラフィックス用 API を呼び出す事によって行っている．その際，プログラマは下層に当たるシステムが何であるか配慮する必要は無い．Java3D では，三次元空間の描画に必要な情報を一つのツリー構造（シーングラフ）として扱うようになっており，大規模なものになってもオブジェクトなどの管理が容易である．このように Java3D は将来的な開発スタイルも見越した上の設計になっている．

2.2 LG3D（ Project Looking Glass）

LG3D は、Java テクノロジをベースとし、3D ウィンドウ処理・表示技術を通じて、より豊かなユーザエクスペリエンスをデスクトップとアプリケーション

にもたらすことを目指している。また、既存のアプリケーションに手を加えること無く、LG3D によってデスクトップに構築された三次元空間で実行できることも考慮している。

現在、このプロジェクトはオープンソースで開発がおこなわれており、多くの人がこれに携わっている。このことにより、デスクトップの進化を，ごく一部の限られた人の手で行うのではなく，多くの人の意見を取り入れられることが可能となり、3DGUI の可能性の探索を精力的に進めることができる。

複数のアプリケーションで単一の仮想 3D 空間で共有するために、クライアント/ サーバシーングラフを作成してある。そのため、アプリケーション実行環境のクリーンな分離ができ，クライアントの挙動不審によるサーバへの悪影響を最小限に留める事ができる。

また、高い生産性を実現するために、Java ベースの API や、モデルローダ、柔軟で強力な 3D レイアウトマネジャを提供している。さらに、豊かなユーザフィードバックの実現のために、容易にアニメーションをおこなえるプリミティブも用意してある。

ウィンドウの整列などデスクトップ操作に限定したマウスジェスチャは用意されている。しかし、開発者が作成したクライアントアプリケーションに、特別なマウスジェスチャを追加できる設計にはなっていない。

3 Cosmo Scheduler D の仕様

Cosmo Scheduler D は LG3D 上で動くスケジュール管理 3D アプリケーションである．スケジュール帳を太陽系に見立てた美しい GUI とシンプルなインターフェースを特徴としている．ユーザは一人につき１つの惑星系を所有し，１つの予定は１つの星として 3D 空間上に浮かべられる．星は時間の流れを表す惑星軌道上に，その予定の設定日（時間）によって配置される．また，予定の内容によってその星の形や大きさも変化する．現在時刻から遠く離れた予定は，ユーザから距離的にも遠く表示されることになる．重要な予定は星が大きくなっているので，重要でない予定よりも目立つことになり，ユーザは重要な予定をいつも気にかけていられる．注目している日付を詳しく見ながら，同時に前後の予定を概観できるということも特徴である．また，予定に関連したファイル，書類や資料などをその星の周りに衛星として浮かべることができる．衛星はそのファイルに対応したランチャーアイコンで，衛星をクリックすることでアクセスすることができるようになっている．

惑星軌道の形は，標準では放物線を下から眺めたような形状をしている．放物線の頂点が現在時刻を表しており，時間が現在時刻から離れるにつれて放物線の軸から離れた位置に配置される．時間が進むにつれて，オブジェクトは放物線上を移動していく．この形状によって星までの時間的距離を，物理的な距離として視覚的に把握することができるようになっている．また，予定の要素であるカテゴリー別に複数軌道を用意することができ，各予定はそれぞれの軌道上に浮かべられる．これらの軌道は，螺旋形状にも変更できる（図 2 ）．螺旋形状によって，各予定の周期性を視覚的に見ることができる．螺旋の周期を変更すれば，毎月，毎週，毎日，隔週などの任意の視点によるスケジュール閲覧が可能になる．

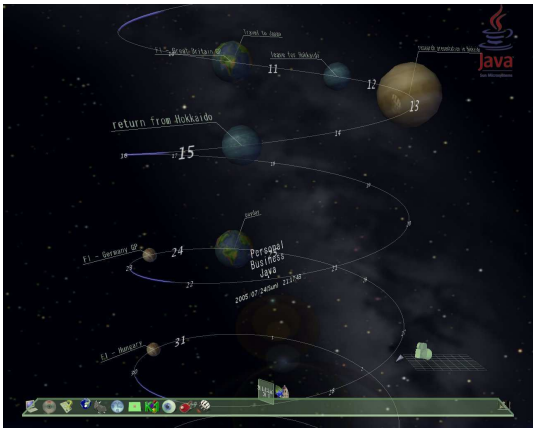

図 2: 軌道の螺旋表示

予定追加の操作にもなるべくシンプルなインターフェースを考えた．ポップアップしたパネルに予定名を入力した後は，マウスのみで初期設定を行うことができる．初期設定にはマウスジェスチャー等によるオブジェクトへの直接操作を用いる．まず，パネルをグチャグチャとスクラッチすることで，パネルが小さな星へと丸められる．その星に対して時計回りもしくは半時計回りに円を描くことで，星の大きさを決める．あとは目的の日付に置き，好きなカテゴリーに移動することで追加完了となる．衛星もファイルマネージャから直接ドロップするだけで作成可能となる．

一人のスケジュールが一つの惑星系に対応しているため，他の人のスケジュールとの比較は太陽系同

士の比較で行えるようになっている．この特徴を活かしたネットワーク越しのスケジュール管理機能も開発を行っている．また，各惑星（予定）に対応したワークスペースを用意することで，仕事自体も予定単位で行うという新しいワークスタイルも提案していく予定である．

また，3D 空間上ではカメラワークの自由度も重要になるので，我々はカメラワークを擬似的にアプリケーション単位で実現する仕組みを取り入れてる．アプリケーション単位のカメラワークなので，LG3D 上の他のアプリケーションに影響することはない．これは LG3D のように複数の 3D アプリケーションが同居するような環境で重要になってくると考えられる．

4 Cosmo Scheduler D の実装

予定クラス（Plan）の要素は以下の通りである．

- 予定名
- 予定日付，時刻
- 予定準備開始日付
- 予定終了日付，時刻
- カテゴリー
- 関連ファイル記述
- 重要度
- テクスチャファイル名
- 予定準備作業量（所要時間）

この要素のうち，Cosmo Scheduler D の実装のために特別に用意した要素を以下に説明する．優先度は星の大きさに反映される．優先度が大きくなるにつれて星も大きくなる．関連ファイル記述には，予定に関するファイルが指定してある．システムはそのファイルの種類を判別し，予定星の周りに浮かべる．ユーザは予定単位でファイルにアクセスすることが可能になる．星にはデフォルトで数種類の模様を用意してあるが，ユーザが特別にその予定を区別したい場合，好きな画像ファイルを星に貼り付けることができる．その画像ファイル名を「テクスチャファイル名」に保持している．

例） 予定準備開始日付は，その予定を準備開始する日程を表す．予定終了日付はその予定が期間を持つ場合に使用される．例として「3月１０日の学会Ａで発表」という予定を挙げる．学会発表の資料等を用意し始めるべき日が１週間前だとすると，予定準備開始日付は３月３日となる．学会Ａが３日間あるならば，予定終了日付は３月１２日となる．予定準備作業量が合計３５時間である場合，１日当たり５時間準備作業をすべきだとわかる．

Cosmo Scheduler D は以上の要素を利用して，予定を惑星に見立ててスケジュールを表示している．

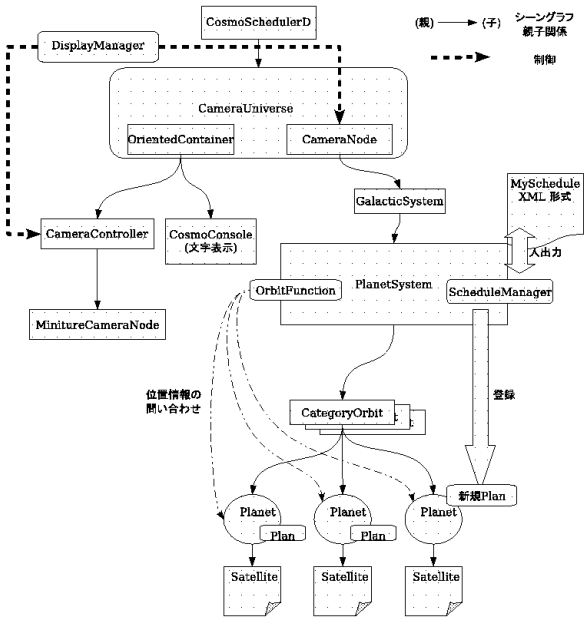

図 3: Cosmo Scheduler D のクラス概図

4.1 Orbiter クラス

予定をオブジェクトとして表示する実体が Orbiter クラスと Planet クラスである．Orbiter クラスと予定は１対１対応しており，予定の数だけ Orbiter ができる．Planet クラスは Orbiter クラスの具象クラスである．Planet クラスは予定を星として表示する機能を持っている．Orbiter クラスは惑星系軌道上へのオブジェクト配置と Plan クラスの操作の基本機能を有している．今後の拡張では，期間を持つ予定は流星として表示するような，予定の内容に見合った形状の変更を行う予定である．そのため，Orbiter クラスを抽象クラスとして実装している．Satelite クラスは Orbiter クラスが持つ関連ファイルのアイ

コンとなるクラスである．Plan の関連ファイル記述の内容に従って作られる．Satelite をクリックすることで，関連ファイルを開くことができる．現在の実装では画像ファイルをビューワで開けるようになっている．

4.2 PlanetSystem クラス

多数作られる Planet クラスを統括するのが PlanetSystem クラスである．一つのスケジュールファイルつき一つの PlanetSystem となる．スケジュールファイルは PlanetSystem が持つ ScheduleManager クラスが管理する．ScheduleManager はスケジュールが記述されたファイルから読み込み，Plan クラスを実体化，PlanetSystem へと登録する．登録された Planet クラスはそのカテゴリーによって，各 CategoryOrbit に振り分けられ，シーングラフに付加される．登録されたメンバーのスケジュールファイルは別の PlanetSystem として実体化され，各 PlanetSystem クラスは GalacticSystem クラスに所属する．

4.3 DisplayManager クラス

DisplayManager はカメラワークやそれに連動した表示切替を司るクラスである．CameraUniverse はシーングラフ上で二つの枝別れを作る．一方がカメラワークに影響される枝で，もう一方は影響されない．カメラの移動や向き変更に影響されないため，その枝の下に各種文字表示などのオブジェクトを付加している．CameraNode は実際のカメラの機能を持つクラスであり，注視点を中心にしたカメラワークを実現する．注視点の Y 軸周りの角度 yaw，X 軸周りの角度 angle，注視点との距離 distance，注視点の位置 pointOfRegard の要素を持っており，それらの値をセットすると時間 duration の値の間にスムースに保管してアニメーションし，各値が変更される．CameraUniverse にはこの CameraNode クラスもしくはその継承クラスを設定することになる．

ユーザは CameraController を使って，DisplayManager はセットされた CameraNode を視覚的に操作することができる．CameraNOde の角度と MinitureCamera は相互に影響し，ユーザの操作をフィードバックする．カメラの状態を視覚的に把握することと同時に，簡単な操作でのカメラワークを実現している．また，マウスホイールにより軌道に沿って移動することができる．CameraNode と CameraUniverse は CosmoAPI の中核として更なる機能充実を図っていく予定である．

4.4 CosmoConsole クラス

3D 空間上に自由な位置に浮かべられたオブジェクトから文字をポップアップさせることは，Widget ベースの 2DGUI に対して複雑な処理が必要となる．オブジェクトの座標系に文字を浮かべるだけでは，遠距離にあるオブジェクトの文字は見えない程小さくなってしまうことになる．その問題を解消できるように，カメラから見た視点でそのオブジェクトと丁度重なる位置に文字を表示する機能を CosmoConsole として実装した．

4.5 OrbitFunction クラス

3DGUI ではオブジェクトの位置による情報表示の自由度が上がるため，我々は 3DGUI の利点として活用することを目指した．その実現の一部として実装したものが OrbitFunction である．インターフェース OrbitFunction の実装クラスは各種関数を表している．その例が Parabola クラスと Spiral クラスであり，それぞれ 2 次関数と螺旋を表している．先述した Orbiter クラスはこの関数上に配置される．今回はスケジュールの時間（ミリ秒）を入力として与え，その位置を Vector3f クラスとして返すようにしてある．この関数を利用して，軌道のラインオブジェクトや日付のパネル配置，曜日の装飾などのオブジェクトの形状も決定している．この関数を新たに開発すれば，各種パラメータの値を効果的に表すことができると我々は考えている．また，カテゴリーによって軌道ラインを複数用意しており，軌道上での位置ではなく，軌道自体の位置によっても情報を表示している．

まとめると，軌道の位置によって予定の時間，星の大きさにより予定の重要度，軌道の位置によって予定のカテゴリー，さらに予定の色によって予定の特徴付けを行える．

4.6 AppGestureModuleBase クラス

LG3D には，ウィンドウ自体の操作方法としてマウスジェスチャーを取り入れてある．3D アプリケーションでは，ユーザはよりオブジェクトを直接操作する感覚が強いため，マウスジェスチャーとの親和

性も良いはずであると考えた．そのため，マウスジェスチャー機能をアプリケーション側でも利用できる抽象クラス AppGestureModuleBase を作成した．オブジェクトからのマウスドラッグによりマウスの軌跡に沿ってラインが引かれる．マウスカーソルの座標を対象オブジェクトの座標系に変換し，その座標系にラインノードを addChild することでラインを描画している．マウスボタンがリリースされるとその軌跡からジェスチャーを解析し，そのオブジェクトにジェスチャーの内容を与えることができる．このことによりオブジェクト毎にジェスチャーによる動作を変更することもできる．LG3DAPI のジェスチャー解析クラスを拡張して作成している．

5 評価

本章では CosmoAPI の評価を行なう．評価には CosmoAPI を使って作られた Cosmo Scheduler D と同じく lg3d-incubator[4] に登録されているいくつかのアプリケーションと比較することで，CosmoAPI の適用できる範囲がどの程度存在するかとその効果について検証する．

なお，lg3d-incubator とは LG3D 上で動作するアプリケーションをオープンソースで開発していこうとするプロジェクトである．ここに登録されているものは，3D オブジェクトで構成された立体的なアプリケーションである．

5.1 CosmoAPI の適用範囲

lg3d-incubator に登録されているそれぞれのアプリケーションは，どのアプリケーションにも共通して使用可能な処理を独自に実装している．例えば，Cosmo Scheduler D と Clock（図 4 ）では，以下に挙げるいくつかの処理を共有できる．

文字列の表示 ユーザに情報を伝える手段として有用な文字を LG3D で表示する場合は，文字を書いたテクスチャを動的に作成し，パネルなどのオブジェクトに貼る操作が必要になる．Clock の場合，デジタル時計の表示、アナログの文字盤等に使用される．

軌道に沿った移動 Clock では，デジタル，アナログを切替える際のアニメーションは回転しながら遠ざかり，切替えて元の場所に近付いてくるアニメーション表現を行っている．この処理は，Cosmo Scheduler D と同じく，一定期間，どのように移動するといったスレッドを作成してアニメーションを実行している．

このような部分は，各アプリケーションで共通した処理が大半を占めており，CosmoAPI を用いることで簡単化と拡張性を確保することができる．

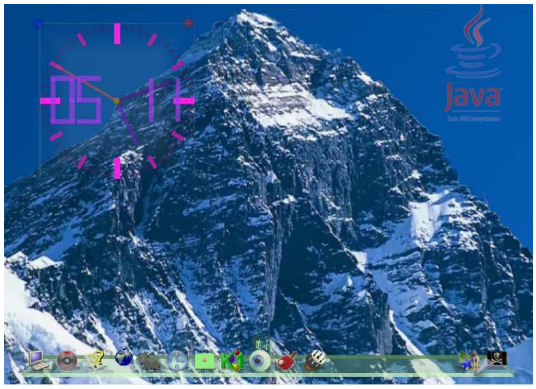

図 4: Clock:xclock の LG3D 版．壁掛け時計のようにデスクトップ空間に配置される

また，Cosmo Scheduler D で使用している，マウスのある位置にオブジェクトを表示する機能は，2DGUI アプリケーションにおけるポップアップ表示と同様の表現が可能になる．先の Clock に適用すれば，デジタル表示，アナログ表示の切替えボタンにマウスカーソルがあった時に，そのコンポーネントをクリックにより表示切替えが可能ということの説明を表示するように拡張できる．

このように，CosmoAPI の利用により，コード量の減少や新たな機能拡張の容易性を望むことができため，今後の 3DGUI アプリケーションの開発において CosmoAPI は有効である．

5.2 CosmoAPI の効果

3DGUI の利点のひとつとして，ユーザに見せたい部分を強調しながら，その周りを概観として表示できるということが挙げられる．特に概観には 3 次元的な表現を加えることで全体像を理解しやすくなる．lg3d-incubator に登録されているアプリケーションの中から，こうしたことを行っているアプリケーションを選択すると Cosmo Scheduler D と Zoetrope（図 5 ）があてはまる．

Cosmo Scheduler D では，時間を横軸の数直線で表すだけでなく，奥行き方向にも 2 次関数で遠ざかっ

図 5: zoetrope:スライドビューア．サムネイルが円筒状に並んでいる．

図 6: Knowledge Web Demo #1

ている．注目している箇所はカメラに一番近い現在の予定が表示されている場所で，そこから離れる程小さく表示されている．このことにより，過去や未来の予定の時間的距離を，3 次元特有の遠近法を利用して，物理的な距離感覚としてユーザにフィードバックしている．この点において，Cosmo Scheduler D は 3 次表現の特性をうまく活かしたアプリケーションのひとつである．

一方，Zoetrope も 3 次元の要素を巧みに利用している．Zoetrope とは覗き穴から筒内の絵が動いて見える回転仕掛けの玩具のことであり，日本語では回転のぞき絵という意味である．Zoetrope はこの回転のぞき絵をメタファとしたスライドビューアである．このアプリケーションでは，複数の画像のサムネイルを立体的な円筒上に配置するという 3 次元を活かす工夫がなされており，多くの画像の中から一つを注視することが自然に表現されている．

3 次元表現を前提としたアプリケーションとしては，Knowledge Web Demo #1 （図 6 ，[5]）が登録されている．多くの事物の複雑な関係全体を球として 3 次元で表現しているが，情報量の多さから注視すべき部分が分かりくく，過密な印象となっている．

その他のアプリケーションでは，主にアニメーションを行う部分に 3 次元独特の表現を使用している．例えば，lg3d-incubator の中では最も完成度の高いアプリケーションの一つである電子メール閲覧ブラウザ BlackGoat（図 7 ）は，コンポーネントを選択時，マウスカーソルがコンポーネント上に移動した際の強調表示，新規ウィンドウを表示時などのアニメーションに回転や拡大縮小，半透明化のような 3 次元的表現手法を使用している．しかし，こうしたアニメーション表現の効果については今後検証すべきである．

図 7: BlackGoat:LG3D 初の MUA．アニメーションを駆使した楽しげな使用感がある．

以上のことから，3 次元表現を効果的に利用して，その利点を充分に活かしているアプリケーションはまだ多くはない．

6 まとめ

今後の 3D デスクトップ市場を促すために，開発に必要な API と見本となる 3D アプリケーションを用意した．用意した API を使えば，LG3D の API だけでアプリケーションを開発するよりも作業が簡略化でき，表現や拡張性も増す．

また，lg3d-incubator の各アプリケーションを含め，3D アプリケーションはまだ発展途上であり，3 次元表現することの効果とその利点をまだ模索段階だと思われる．今後は様々な思考錯誤を繰り返し，どのようなアプリケーションで 3 次元表現として何が

必要であり，どのように表現するかを検討していく必要がある．

なお，本 API を利用して作成した Cosmo Scheduler D は，2005 年 6 月 27-30 日にサンフランシスコで開かれた JavaOne において高い評価を得て，Duke's Choice Award を受賞している．

謝辞

本研究に関してご議論いただいている Sun Microsystems の川原英哉氏，JavaOne 2005 にてご議論いただいた lg3d コミュニティの方々に感謝いたします．また日頃からご議論いただいている九州工業大学 Network Design Research Center の方々に感謝します．本研究は，文部科学省科学研究費補助金（課題番号 16016271）および IPA 未踏ソフトウェア創造事業の支援により遂行されました．

参考文献

[1] LG3D プロジェクトホームページ
https://lg3d-core.dev.java.net/

[2] XOrg http://www.x.org/

[3] Java3D https://java3d.dev.java.net/

[4] lg3d-incubator プロジェクトホームページ
https://lg3d-incubator.dev.java.net/

[5] KnowledgeWeb Project http://www.k-web.org/