試験の目的

Unity 認定
プログラマー

役割

Unity プログラミングのプロフェッショナルは、Unity を使用してインタラクティブなコンテンツを開発します。Unity プログラマーは、開発チームの他のメンバー（オーディオやアートのプロフェッショナルなど）と協力し、Unity Editor の機能とソフトウェア開発チームの他のメンバーが作成したビジュアルアセットやオーディオアセットを使って、アプリケーションに生き生きとした世界観を作り出します。Unity プログラマーは、困難なコーディングの問題解決に精通したジェネラリストです。ジェネラリストの役割は、アートアセットの統合、ユーザーインターフェースのコーディング、ユーザーインタラクションやゲームシステムルールのスクリプト作成、アプリケーションの状態ロジックの実装、物理シミュレーション、コードのデバッグ、パフォーマンスの最適化など、考えられるさまざまな技術的タスクに貢献することです。

Unity 認定プログラマーは、さまざまな業界でプログラミング職を探している、初級から中級レベルのプログラマーおよび大学や専門学校の卒業予定者を対象としたプロフェッショナル認定資格です。この認定資格があれば、採用担当者に以下を示すことができます。

- プロフェッショナルなソフトウェア開発プロセスの中でプログラミングの手腕を発揮し、Unity Engine を使用したアプリケーションの作成と保守を行うことができる
- 技術的なプロセスに適性があり、論理指向で優れた能力を持っている
- ルーチンから中級レベルまでのプログラミングタスクを独力で処理し、上級エンジニアと組んで複雑な技術上の課題に対処するように委託できる

この役割に対応する職種

- ゲームプレイプログラマー
- ソフトウェアエンジニア
- ソフトウェア開発者
- Unity 開発者
- モバイルアプリケーション開発者

前提条件

この認定資格は、ゲームプログラミング、コンピューターサイエンス、または関連分野の大学を最近卒業したプログラマー、2 年以上の大学相当の教育を受けているか、プログラミングの実務経験を持つ自立した学習者、あるいは職務として Unity を使用した経験を持つ初心者から中堅までのプロフェッショナルを対象として作成されました。受験者は、個人またはクロスファンクショナルチームの一員として、Unity を使用してインタラクティブアプリケーションをプログラミングし、プロトタイプや技術実証を完成させた実務経験を有している必要があります。

前提条件の内容 :

- Unity を使用したビデオゲームまたは 3D インタラクティブコンテンツのプログラミングの実務経験 2 年以上
- C# を含むコンピュータープログラミングの実務経験 2 年以上
- 構想から完成までソフトウェア開発ライフサイクル全般に携わった経験
- ゲーム開発、インタラクティブエンターテイメント、デザインビジュアライゼーションなど、Unity を使用したソフトウェア開発のための専門的なアプリケーションの知識
- キャラクターや環境の設定など、Unity でのビジュアル /3D アセットおよびアニメーションパイプラインについての基本的な知識
- 単体テストとバージョン管理など、プロフェッショナルチームによるソフトウェア開発の手法に対する理解
- コラボレーション、収益化、ライブオペレーション、マルチプレイヤーのための Unity サービスに関する知識
- 線形代数や行列演算など、3D インタラクティブ開発に不可欠な数学の知識

注 : この認定資格は、Unity バージョン 2017.3 を元に作成されました。

中心的なスキル

この分野の中心的なスキルは、構想から製品の完成、そしてその先までプロジェクトの技術的な遂行に貢献できることに重点を置いたものです。

中核となるインタラクションのプログラミング

- ゲームオブジェクトの動作と物理特性を実装して構成する
- 入力およびコントロールを実装して構成する
- カメラビューおよびカメラ動作を実装して構成する

アートパイプラインでの操作

- マテリアル、テクスチャ、シェーダーを理解し、Unity のレンダリング API を操作するスクリプトを作成する
- ライティングを理解し、Unity のライティング API を操作するスクリプトを作成する
- 2D アニメーションおよび 3D アニメーションを理解し、Unity のアニメーション API を操作するスクリプトを作成する
- パーティクルシステムおよびパーティクルエフェクトを理解し、Unity のパーティクルシステム API を操作するスクリプトを作成する

アプリケーションシステムの開発

- メニューシステム、UI ナビゲーション、アプリケーション設定など、アプリケーションインターフェースフローのスクリプトを解釈する
- キャラクター作成システム、インベントリ、ストアフロント、アプリ内課金など、ユーザーが制御するカスタマイズのスクリプトを解釈する
- Unity Analytics や PlayerPrefs などの技術を使用したスコア、プレイヤーレベル、ゲーム内経済など、ユーザーの進行度に関する機能のスクリプトを分析する
- ヘッドアップディスプレイ (HUD)、ミニマップ、広告などの 2D オーバーレイのスクリプトを分析する
- アプリケーションデータとユーザーデータを保存および取得するためのスクリプトを特定する
- ネットワーキング機能とマルチプレイヤー機能の影響を認識して評価する

シーンと環境デザインのプログラミング

- オーディオアセットを実装するためのスクリプトを決定する
- ゲームオブジェクトのインスタンス化、破壊、管理を実装する方法を特定する
- Unity ナビゲーションシステムによる経路探索のスクリプトを決定する

パフォーマンスとプラットフォームに合わせた最適化

- Unity Profiler などのツールを使用してエラーやパフォーマンスの問題を評価する
- 特定のビルドプラットフォームやハードウェア構成の要件に対処するための最適化を確認する
- XR プラットフォーム向けの一般的な UI アフォーダンスおよび最適化を決定する

プロフェッショナルなソフトウェア開発チームへの参加

- Unity Collaborate などの技術を使用して、バージョン管理の使用および影響に関連した概念を理解する
- Unity Profiler、従来のデバッグ手法やテスト手法など、ソフトウェア開発プロセスにおける開発者テストとその影響に関する知識を明らかにする
- モジュール性、可読性、再利用性を実現するためのスクリプトを構成する手法を理解する

認定試験のトピック

中核となるインタラクションのプログラミング

- ゲームオブジェクトおよび環境の動作とインタラクションを実装する
- 入力とコントロールを実装する方法を特定する
- カメラビューとカメラ動作を実装する方法を特定する

アートパイプラインでの操作

- マテリアル、テクスチャ、シェーダーの知識 – Unity レンダリング API
- ライティングの知識 – Unity ライティング API
- 2D アニメーションおよび 3D アニメーションの知識 – Unity アニメーション API
- パーティクルシステムの知識 – Unity パーティクル API

アプリケーションシステムの開発

- メニューシステム、UI ナビゲーション、アプリケーション設定などのアプリケーションインターフェースフロー
- キャラクター作成システム、インベントリ、ストアフロント、アプリ内課金など、ユーザーが制御するカスタマイズ
- Unity Analytics などのツールを使用したスコア、プレイヤーレベル、ゲーム内経済など、ユーザーの進行度に関する機能を実装する
- ヘッドアップディスプレイ (HUD)、ミニマップ、広告などの 2D オーバーレイを実装する
- アプリケーションデータとユーザーデータの保存と取得
- ネットワーキング機能とマルチプレイヤー機能の価値と影響を認識する

シーンと環境デザインのプログラミング

- オーディオアセットを実装するためのスクリプトを決定する
- ゲームオブジェクトのインスタンス化、破壊、管理を実装する方法を特定する
- Unity ナビゲーションシステムによる経路探索のスクリプトを決定する

パフォーマンスとプラットフォームに合わせた最適化

- Unity Profiler などのツールを使用してエラーやパフォーマンスの問題を評価する
- 特定のビルドプラットフォームやハードウェア構成の要件に対処するための最適化を確認する
- XR プラットフォーム向けの一般的な UI アフォーダンスおよび最適化を決定する

ソフトウェア開発チームへの参加

- バージョン管理 : Unity Collaborate などのツールの影響と使用
- テストおよびソフトウェア開発プロセスに対するその影響
- モジュール性、可読性、再利用性を実現するためのスクリプトを構成する手法の理解

サンプル問題

問題 1

あるプログラマーが UI メニューシステムを実装することになりました。各メニューは 1 つの UI パネルと 1 つ以上の UI ボタンで構成され、それらはすべて、UI Canvas オブジェクトを親とします。UI メニューシステム全体は、付加的に読み込まれる別のシーンに作成します。

パネルとボタンのアートスタイル（色、テクスチャ、ボタン遷移タイプなど）は一貫させる必要がありますが、アートディレクターはまだこれらの最終決定を下していません。アートディレクターは、プログラマーによる UI 作成と並行してこれらの設定に取り組みたいと考えています。アートディレクターによる変更は、シーン内の新しいオブジェクトと既存のオブジェクトすべてに反映されます。

プログラマーが Unity の機能を使用して実用的なメニューシステムを簡単に作成しながら、アートディレクターが同時に（および個別に）ルック＆フィールに取り組めるようにする要件を満たす方法として最適なものを選択してください。

A `UI.Button` と `UI.Panel` のサブクラスを作成し、ルック＆フィールの値をプログラムで設定する

B 新しいボタンとパネルのマテリアルを作成し、シーンのすべてのボタンとパネルに割り当てる

C ボタンとパネルにプレハブを使用し、アートディレクターにプレハブを修正してもらう

D アートディレクターの入力内容に基づいてシーンファイル内の値を検索 / 置換するスクリプトを作成する

問題 2

車両基地の中に複数の線路が並行して走っている、3D のエンドレスランナーゲームがあります。プレイヤーは線路上を常に前に向かって走っており、対向列車が来た場合は、飛び越えたり隣の線路に飛び移ったりして避ける必要があります。
新しい列車は、線路に追加されるたびに、その線路上のその他すべての列車の後ろに追加されます。ところが、対向列車のスピードはさまざまであるため、たまに列車どうしが重なることがあります。これは、修正する必要があります。

新しい列車が同じ線路上の既存の電車と重ならないようにするのに最も効率的な方法はどれですか？

A 線路上に列車を生成するときに、新しい列車と線路に最後に配置した列車の速度、および線路に最後に配置した列車がプレイヤーを通過したときに消滅する位置を使用して、この問題を回避する生成位置を決定する

B 動いている列車の前面から前方に向かってレイキャストし、レイキャストでヒットした列車をより速い列車速度を使用して前に押し出す

C 線路上に列車を生成する場合、その列車に Rigidbody を追加し、フォースを使って列車を動かす

D 線路上に列車を生成する場合、長さが列車の速度に比例した BoxCast を使用して、列車がカメラを通過するまで他の列車と衝突しないようにする

問題 3

プログラマーは暗くて雰囲気のある部屋を作成しており、壁、床、天井に不気味な影を映し出すちらちらと揺らめくたいまつを作成する必要があります。プログラマーは、たいまつにアタッチされた MonoBehaviour について、次の関数を記述しています。

```
void Start()
{
    Light light = GetComponent<Light>();
    light.lightMapBakeType = LightMapBakeType.Mixed;
    light.type = LightType.Area;
    light.shadows = LightShadows.Soft;
    light.range = 5f;
}
void Update()
{
    GetComponent<Light>().intensity = Mathf.PerlinNoise(Time.time, 0);
}
```

実行時、たいまつは光も影も作りません。Unity Editor で、ライトはデフォルト値に設定されています。

このコードを仕様どおりに動作させるには、プログラマーは何を変更する必要がありますか？

A `light.lightBakeType` を `LightmapBakeType.Realtime` に設定する

B `light.range` を `10` に設定する

C `light.shadows` を `LightShadows.Hard` に設定する

D `light.type` を `LightType.Point` に設定する

問題 4

プログラマーは採掘シミュレーションゲームを開発しています。このゲームでは、プレイヤーが鉱物を探し求めて地面を掘り進めることができます。サイトの 1 つで、プレイヤーは既存の洞窟系と交差するトンネルを作成することができます。デザインドキュメントでは、現在の洞窟と新しいトンネルの両方で再生されるオーディオに残響が生じるよう指定されています。プログラマーは、ユーザーが最も近い洞窟の ReverbZone に常に属しているようにする必要があります。

上記の要件を満たすために、プログラマーは AudioReverbZone のプロパティをどのように操作する必要がありますか？

A 新しい領域に合わせてリフレクションを大きくする

B 両方の ReverbZone の maxDistance を大きくして、これらが新しい連結領域内で接触するようにする

C 新しい領域に合わせて残響を大きくする

D 新しい領域の decayTime を大きくする

問題 5

プログラマーが読み込み関数の記述中に次のコンパイルエラーが発生しました。

```
error CS1624: The body of `CustomAnalytics.LevelLoading()' cannot be
an iterator block because `void' is not an iterator interface type
```

```
void LevelLoading(){
    AsyncOperation async = SceneManager.LoadSceneAsync("Level_01");
    while (!async.isDone)
    {
        yield return null;
    }
}
```

このエラーを修正するためにプログラマーはどうすればよいですか？

A `yield return null` を `yield return WaitForSeconds(0)` に変更する

B `void LevelLoading()` を `IEnumerator LevelLoading` に変更する

C `SceneManager.LoadSceneAsync("Level_01")` を `Application.LoadLevelAdditiveAsync("Level_01")` に変更する

D `while (!async.isDone)` を `while (!async.allowSceneActivation)` に変更する

問題 6

ドライビングゲームの入力システムは、水平入力軸でステアリングを制御するようにマッピングされています。テスト中、一部のジョイスティックデバイスで、スティックが中央にあってもステアリング入力が登録されることがわかりました。

この問題を解決するために、入力システムの軸にどのような変更を加える必要がありますか？

A Gravity を大きくする

B Snap を true に設定する

C Deadzone を大きくする

D Sensitivity を小さくする

問題 7

エイリアンのいる惑星を舞台としたアドベンチャーゲームで、プレイヤーはさまざまな生命体を倒す必要があります。プレイヤーが生命体を倒すたびにスコアが加算されます。デザインドキュメントには、プレイヤーが別のセッションや別のデバイスでプレイした場合でも後でスコアを取り戻せるようにするために、スコアをプレイヤーのアカウントにリンクする必要があると記述されています。

プログラマーがスコアデータを保存するための最も信頼性の高い方法はどれですか？

A スコアデータを保持するゲームオブジェクトで `DontDestroyOnLoad()` を使用して、アプリケーションの終了直前にサーバーにデータをアップロードする

B スコアが更新され、アプリケーションの終了直前にサーバーにアップロードされるたびに `PlayerPrefs` にそのスコアを保存する

C 静的な値を使用してスコアデータを保存し、次のプレイセッションで使用できるようにする

D データのシリアル化を使用してスコアデータを永続的に保存し、サーバーにアップロードする

正解 : C、A、D、B、B、C、D