

✓ フォワードレンダリングの説明として、適切なものを下記から選びなさい。 25/25

- ☒ モデルをドローするタイミングでライティングの計算を行う手法。 ✓
- ☐ モデルをドローするタイミングでG-Bufferを作成して、遅れてライティングの計算を行う手法。

✓ ディファードレンダリングの説明として、適切なものを下記から選びなさい。 25/25

- ☐ モデルをドローするタイミングでライティングの計算を行う手法。
- ☒ モデルをドローするタイミングでG-Bufferを作成して、遅れてライティングの計算を行う手法。 ✓

✓ ディファードレンダリングのメリットとして、適切なものを下記から選びなさい。 25/25

- ☒ ライティングの計算回数が、モデルの描画順番に依存せず、固定化することができる。 ✓
- ☐ 半透明オブジェクトを正しく描画することが容易になる。
- ☐ フォワードレンダリングと比べてメモリ使用量を削減することができる。

✓ ディファードレンダリングでモデルのドローのタイミングで作られる、テクスチャの名称として、適切なものを下記から選びなさい。 25/25

- ☐ SSAO
- ☐ SSR
- ☒ G-Buffer ✓
- ☐ 被写界深度

- ✓ ディファードレンダリングで鏡面反射を実装する際にG-Bufferに追加 100/100
する必要があるものとして適切なものを下記から選びなさい。

- ☒ ピクセルのワールド座標を記憶するためのレンダリングターゲット ✓
- ☐ ピクセルの速度を記憶するためのレンダリングターゲット
- ☐ ピクセルにかかっている重力を記憶するためのレンダリングターゲット

- ✓ アニメ調のセルルックシェーダーでは、ピクセルの深度値や法線などを 40/40
利用して、物体と物体の境界線を表す①_____を描画することが多
い。下線部 1 に当てはまる語句として、適切なものを下記から選びなさい。

- ☒ 輪郭線 ✓
- ☐ ライト
- ☐ 光沢

- ✓ シンプレックスノイズアルゴリズムの説明として、適切なものを下記か 30/30
ら選びなさい。

- ☒ パーリンノイズの改良版のアルゴリズムで、連続性のある乱数生成アルゴリズムである。 ✓
- ☐ 線形合同法の改良版のアルゴリズムで、連続性のない乱数生成アルゴリズムである。

- ✓ ディザパターンを使用して、3Dオブジェクトを穴抜きで表示する手法 30/30
の名称として、適切なものを下記から選びなさい。

- ☐ 輪郭線抽出
- ☐ 深度バッファ
- ☒ ディザリング ✓
- ☐ アンチエイリアス

✓ エネルギー保存の法則の説明として、適切なものを下記から選びなさい。 20/20

- ☒ エネルギーの総量は一定であるという法則 ✓
- ☐ 光が入射してくる方向と、射出の方向を入れ替えても、光の射出量の結果が変わらないという法則

✓ ヘルムホルツの相反性の説明として、適切なものを下記から選びなさい。 20/20

- ☐ エネルギーの総量は一定であるという法則
- ☒ 光が入射してくる方向と、射出の方向を入れ替えても、光の射出量の結果が変わらないという法則 ✓

✓ 金属のサーフェイスで発生する鏡面反射の特性として、適切なものを下記から選びなさい。 20/20

- ☐ 入射してきた光源のカラーを鏡面反射光として返す
- ☒ 物体の色を鏡面反射光として返す ✓

✓ 非金属のサーフェイスで発生する鏡面反射の特性として、適切なものを下記から選びなさい。 20/20

- ☒ 入射してきた光源のカラーを鏡面反射光として返す ✓
- ☐ 物体の色を鏡面反射光として返す

✓ 微細表面での物体の凸凹具合を表すPBRパラメータとして、適切なものを下記からすべて選びなさい。 20/20

- ☒ 粗さ ✓
- ☒ 滑らかさ ✓

✓ VSMは深度値の分散を利用して、ソフトシャドウをおこなうアルゴリズム 20/20
ムである。この説明が正しいか、正しくないか下記から選びなさい。

☒ 正しい



☐ 正しくない

✓ VSMを行うために、シャドウマップに書き込む値として、適切なものを 20/20
下記から選びなさい。

☐ 深度値

☒ 深度値と深度値の2乗



✓ シャドウレシーバーでは、深度値の分散を利用して、①_____ 20/20
を解くことで、影が落ちる確率を計算する。下線部①に当てはまる語句
として、適切なものを下記から選びなさい。

☐ 三平方の定理

☐ 解の方程式

☒ チェビシェフの不等式



☐ 内積

✓ VSMのアルゴリズムの流れとして、適切なものを下記から選びなさい。 20/20

☒ ①シャドウキャストの深度値と深度値の2乗をシャドウマップに書き込む → ✓
②シャドウマップにブラーをかける。→③シャドウレシーバーでチェビシェフ
の不等式を利用して、影を落とす確率を計算する。

☐ ①シャドウレシーバーでチェビシェフの不等式を利用して、影を落とす確率を計
算する。→②シャドウキャストの深度値と深度値の2乗をシャドウマップに書
き込む → ③シャドウマップにブラーをかける。

- ✓ VSMの欠点として、シャドウキャストが重なる部分で、急激な深度値 20/20 の変化が起きると、深度値が分散することになるため、①_____が発生する。

☐ マッハバンド

☐ 桁落ち

☒ 光のにじみ ✓

☐ 情報落ち

- ✓ カスケードシャドウマップの説明として、もっとも適切なものを下記か 25/25 ら選びなさい。

☒ 複数枚のシャドウマップを用意して、カメラからの距離に応じて、シャドウマ ✓
ップを切り替えるアルゴリズムである。

☐ シャドウマップに書き込まれている深度値の分散を利用して、ソフトシャドウを
かけるアルゴリズムである

- ✓ カスケードシャドウ法で、最も高解像度なシャドウマップを割り当てる 25/25
エリアの説明として、適切なものを下記から選びなさい。

☒ カメラからもっとも近いエリア ✓

☐ プレイヤーキャラクターからもっとも近いエリア

☐ カメラからもっとも遠いエリア

- ✓ カスケードシャドウ法では、各エリアに含まれているオブジェクトを適 25/25
切なシャドウマップに必ず描画する必要があるため、クローン行列など
を利用するなど、特殊な座標変換を行う必要がある。この説明が正しい
か正しくないか下記から選びなさい。

☒ 正しい ✓

☐ 正しくない

✓ カスケードシャドウ法のアルゴリズムの流れとして、適切なものを下記 25/25 から選びなさい。

☐ ①各シャドウマップにシャドウキャスターを描画する→②各エリアを内包する視錐台の8頂点を計算する→③クロップ行列を計算する→④シャドウマップを利用して影を落とす。

☒ ①各エリアを内包する視錐台の8頂点を計算する→②クロップ行列を計算する→③各シャドウマップにシャドウキャスターを描画する→④シャドウマップを利用して影を落とす。 ✓

✓ GPGPUの説明として、適切なものを下記から選びなさい。 20/20

☐ GPUを2枚差しで利用すること

☒ GPUを画像処理以外の、衝突検出処理、物理計算、AIなどの汎用計算に利用すること ✓

✓ DirectX10からGPUで汎用計算を行うことを目的として、追加されたシェーダーの名前として、適切なものを下記から選びなさい。 20/20

☐ 頂点シェーダー

☐ ピクセルシェーダー

☒ コンピュートシェーダー ✓

☐ ジオメトリシェーダー

✓ HLSLシェーダーで利用され、C言語の構造体の配列のように扱うことができるシェーダーリソースの名称として、適切なものを下記から選びなさい。 20/20

☒ ストラクチャードバッファ ✓

☐ サンプラステート

☐ テクスチャ

☐ レンダリングターゲット

- ✓ 次のシェーダーリソースビューの中から、シェーダー側から読み書き可能なビューを選びなさい。 20/20

☐ シェーダーリソースビュー

☒ アンオーダーアクセスビュー



- ✓ 次のシェーダーリソースビューの中から、シェーダー側から読み込み専用のビューを選びなさい。 20/20

☒ シェーダーリソースビュー



☐ アンオーダーアクセスビュー

- ✓ 次の画像のコンピュートシェーダーがディスパッチされた。この時、スレッドグループあたりのスレッド数がいくつになるか、下記から選びなさい。 20/20

```
[numthreads(2, 2, 1)] //これがスレッドの数!!!  
void CSMain( uint3 DTid : SV_DispatchThreadID)  
{  
    .  
    .  
    .  
}
```

☒ 4



☐ 5

- ✓ スレッドグループの数が10、スレッドグループあたりのスレッドの数が4のとき、合計で起動するスレッドの数として、適切なものを下記から選びなさい。 20/20

☐ 14

☒ 40



- ✓ C++側から次の画像のようにコンピュートシェーダーがディスパッチさ 20/20
れた。この時生成されるスレッドグループの数として、適切なものを下
記から選びなさい。

```
//3×3のスレッドグループが作られる！  
renderContext.Dispatch(3,3,1);
```

☐ 7

☒ 9



- ✓ C++側から次の画像のようにコンピュートシェーダーがディスパッチさ 10/10
れた。この時生成されるスレッドグループの数として、適切なものを下
記から選びなさい。

```
renderContext.Dispatch(4, 5, 1);
```

☐ 10

☒ 20



- ✓ スレッドグループの数が4、スレッドグループあたりのスレッドの数が 10/10
16のとき、合計で起動するスレッドの数として、適切なものを下記から
選びなさい。

☐ 20

☒ 64



- ✓ 次の画像のコンピュータシェーダーがディスパッチされた。この時、スレッドグループ当たりのスレッド数がいくつになるか、下記から選びなさい。 10/10

```
[numthreads(4, 4, 1)]
void CSMain(uint3 DTid : SV_DispatchThreadID)
{
    g_outputData[0].maxScore = 0;
    g_outputData[0].minScore = 100;
}
```

☐ 9

☒ 16 ✓

- ✓ 同一のスレッドグループに属しているスレッドは並列に動作する。この説明が正しいか正しくないか下記から選びなさい。 20/20

☒ 正しい ✓

☐ 正しくない

- ✓ スレッドグループはすべて並列に動作する。この説明が正しいか正しくないか、下記から選びなさい。 20/20

☐ 正しい

☒ 正しくない ✓

- ✓ [numthreads(2, 2, 1)]が指定されているコンピュータシェーダーを2,2,1 ディスパッチした。この時、スレッドに割り当てられるIDの範囲として、適切なものを下記から選びなさい。 10/10

☐ (0, 0, 0) ~ (4, 4, 1)

☒ (0, 0, 0) ~ (3, 3, 0) ✓

✓ TBRの正式名称として、適切なものを下記から選びなさい。

20/20

☐ Team Based Realtime

☒ Tile Based Rendering



☐ Time By Reading

✓ 動的光源の説明として適切なものを下記から選びなさい。

20/20

☒ プログラム実行中にリアルタイムに計算される光源



☐ プログラム実行中には計算されない光源。テクスチャへの焼き付けなど。

✓ 動的光源を増やしてもゲームのパフォーマンスに影響はない。この説明 15/15
が正しいか正しくないか下記から選びなさい。

☐ 正しい

☒ 正しくない



✓ ポイントライトの説明として適切なものを下記から選びなさい。

15/15

☐ 方向と光のカラーを保持しており、位置情報を持たないライト

☒ 方向と光のカラーと位置情報を保持しているライト



- ✓ 次のプログラムはポイントライトがサーフェイスに入射してくる光の方 10/10
向を計算しているプログラムです。下線部①に当てはまるプログラムと
して適切なものを下記から選びなさい。

```
// サーフェイスのワールド座標をworldPos、ポイントライトの座標をptPosとする。  
float3 ligDir = ①_____ (worldPos - ptPos);  
↓
```

☐ dot

☐ max

☐ min

☒ normalize



- ✓ 次のプログラムは、ポイントライトから入射してくる光の方向とサーフ 10/10
ェイスの法線を利用して、ランバート拡散反射を計算しているコードで
す。下線部①に当てはまるプログラムとして、適切なものを下記から選
びなさい。

```
// ライトの方向をligDir、サーフェイスの法線をnormalとする。  
float t = ①_____ (0.0f, dot(-ligDir , normal));  
↓
```

☐ dot

☒ max



☐ min

☐ normalize

- ✓ 次のプログラムは、ポイントライトの影響範囲とポイントライトまでの 10/10 距離を利用して、ポイントライトの影響率を計算しているコードです。
下線部①に当てはまるプログラムとして、適切なものを下記から選びなさい。

// ポイントライトまでの距離をdistance、ポイントライトの影響範囲をrangeとする。↓
float affect = 1.0f - ①_____ (1.0f, distance / range); ↓
↓

☐ dot

☐ max

☒ min



☐ normalize

- ✓ TBDRの正式名称として適切なものを下記から選びなさい。 20/20

☒ Tile Based Deferred Rendering



☐ Tile Based Forward Rendering

☐ Tile Based Deffence Rendering

- ✓ TBDRの説明として、適切なものを下記から選びなさい。 20/20

☐ Forward Renderingの改良版で、スクリーンをタイル状に分割することで、動的光源の計算を高速にするアルゴリズム。

☒ Deffered Renderingの改良版で、スクリーンをタイル状に分割することで、動的光源の計算を高速にするアルゴリズム。 ✓

✓ TBDRのアルゴリズムの流れとして、適切なものを下記から選びなさい。 20/20

- ☐ ① LightCulling ② G-Bufferの作成 ③ ライティングの計算
- ☐ ① ライティングの計算 ② G-Bufferの作成 ③ LightCulling
- ☒ ① G-Bufferの作成 ② LightCulling ③ ライティングの計算 ✓

※Sample16_02 はポイントライトをディファードレンダリングで実装するプログラム。

✓ 【ハンズオンテスト】 Sample_16_02のハンズオンで追加したG-Buffer 20/20
として、適切なものを下記から選びなさい。

- ☒ 射影空間でのZ値を記憶するためのG-Buffer ✓
- ☐ スペキュラ強度を記憶するためのG-Buffer
- ☐ 法線を記憶するためのG-Buffer

✓ 【ハンズオンテスト】 Sample_16_02で追加したG-Bufferのカラーバッファのフォーマットとして、適切なものを下記から選びなさい。 20/20

- ☐ DXGI_FORMAT_R8G8B8A8_UNORM
- ☐ DXGI_FORMAT_R16G16B16A16_FLOAT
- ☒ DXGI_FORMAT_R32_FLOAT ✓

✓ TBRアルゴリズムでのライトカリングをコンピュータシェーダーで実行 10/10
する際に、ディスパッチされるスレッドグループの数の説明として、適切なものを下記から選びなさい。

- ☒ スクリーンを分割するタイルの数 ✓
- ☐ スクリーンに含まれるピクセルの数

- ✓ TBRアルゴリズムでのライトカリングをコンピュータシェーダーで実行 10/10
する際に、ディスパッチされる 1 スレッドグループあたりのスレッドの
数の説明として、適切なものを下記から選びなさい。

☐ スクリーンに含まれる全ピクセルの数

☒ スクリーンを分割するタイルに含まれるスレッドの数



- ✓ ライトカリングで各スレッドが分担して行う処理として、適切なものを 20/20
下記からすべて選びなさい。

☒ タイルに含まれているピクセルの最大深度、最小深度の調査



☒ タイル(小さな視錐台)とポイントライトのあたり判定



- ✓ C言語のminマクロと同様の処理をアトミックに行うことができる、コ 20/20
ンピュータシェーダーの関数として、適切なものを下記から選びなさい。

☒ InterlockdMin()関数



☐ InterlockdMax()関数

- ✓ C言語のmaxマクロと同様の処理をアトミックに行うことができる、コ 20/20
ンピュータシェーダーの関数として、適切なものを下記から選びなさい。

☐ InterlockdMin()関数

☒ InterlockdMax()関数



- ✓ コンピュートシェーダーで利用できる 20/20
GroupMemoryBarrierWithGroupSync()関数の説明として、適切なものを
下記から選びなさい。

- ☒ スレッドグループ内のスレッドの処理の同期をとることができる。 ✓
☐ スレッドグループ同士の同期をとることができる。

- ✓ TBFRの正式名称として、適切なものを下記から選びなさい。 25/25

- ☐ Tile based deferred rendering
☒ Tile based forward rendering ✓

- ✓ TBFRで利用されるZPrepassの説明として、適切なものを下記から選び 25/25
なさい。

- ☒ モデルを描画する前に深度バッファを作成する描画パス。 ✓
☐ モデルを描画した後で深度バッファを作成する描画パス。

-
- ✓ ZPrepassを利用するメリットとして、適切なものを下記から選びなさい。 25/25

- ☒ 軽いシェーダーでZバッファを作ってしまうことで、のちのPBRのような処理 ✓
の重いシェーダーの際に、無駄なピクセルへのシェーディングを行わないよう
にすることができる。
☐ メリットはない

✓ TBFRのアルゴリズムの流れとして、適切なものを下記から選びなさい。 25/25

☐ ①モデル描画 ②ZPrepass ③ライトカリング

☒ ①ZPrepass ②ライトカリング ③モデル描画 ✓

☐ ①モデル描画 ②ライトカリング ③ZPrepass

✓ 大量のオブジェクトを描画する際に、Flyweightパターンで改善される問題として、適切なものを下記から選びなさい。 10/10

☒ メモリ使用量 ✓

☐ CPUパフォーマンス

☐ GPUパフォーマンス

✓ 大量のオブジェクトを描画する際に、インスタンス描画で改善される問題として、適切なものを下記から選びなさい。 10/10

☐ メモリ使用量

☒ CPUパフォーマンス ✓

☐ GPUパフォーマンス

✓ 大量のオブジェクトを描画する際に、インポスター描画で改善される問題として、適切なものを下記から選びなさい。 10/10

☐ メモリ使用量

☐ CPUパフォーマンス

☒ GPUパフォーマンス ✓

✓ Flyweightパターンの説明として、適切なものを下記から選びなさい。 10/10

- ☐ 複数のオブジェクトを一度のドローコールで描画することができる。
- ☐ モデルを一度オフスクリーン描画し、そこで出来上がったテクスチャを利用する描画方法
- ☒ 共有リソースを使いまわすデザインパターン ✓

✓ インスタンスング描画の説明として、適切なものを下記から選びなさい。 10/10

- ☒ 複数のオブジェクトを一度のドローコールで描画することができる。 ✓
- ☐ モデルを一度オフスクリーン描画し、そこで出来上がったテクスチャを利用する描画方法
- ☐ 共有リソースを使いまわすデザインパターン

✓ インボスター描画の説明として、適切なものを下記から選びなさい。 10/10

- ☐ 複数のオブジェクトを一度のドローコールで描画することができる。
- ☒ モデルを一度オフスクリーン描画し、そこで出来上がったテクスチャを利用する描画方法 ✓
- ☐ 共有リソースを使いまわすデザインパターン

✓ DirectX12でインスタンスング描画が可能なAPIとして、適切なものを下記から選びなさい。 10/10

- ☐ DrawIndex
- ☐ DrawAuto
- ☐ Draw
- ☒ DrawInstancing ✓

✓ 頂点シェーダーの引数にインスタンスIDを渡したい場合に指定するセマンティクスとして、適切なものを下記から選びなさい。 10/10

☐ SV_Position

☒ SV_InstanceID ✓

☐ NORMAL

☐ TANGENT

✓ インスタンス描画で複数のインスタンスを別の場所に表示するためには、ワールド行列の配列を送る必要がある。この説明が正しいか、正しくないか下記から選びなさい。 10/10

☒ 正しい ✓

☐ 正しくない