Rapport Technique Exhaustif : Architecture Graphique, Ingénierie des Systèmes Visuels et Implémentation d'Interfaces Tangibles (Étude de Cas : Hearthstone)

1. Introduction : La Philosophie de l'Interface Tangible

La conception d'une interface utilisateur (UI) dans le domaine des jeux de cartes numériques (CCG - Collectible Card Games) ne relève pas simplement de l'agencement esthétique de boutons et de fenêtres. Elle constitue une discipline complexe à la croisée de l'ingénierie logicielle, de la physique simulée et de la psychologie cognitive. Le succès visuel et ergonomique de titres majeurs comme *Hearthstone* repose sur un paradigme fondamental : la "physicalité".[1] Contrairement aux interfaces web ou applicatives traditionnelles qui sont "plates" et abstraites, l'interface de *Hearthstone* est conçue comme un objet physique, une "boîte" tridimensionnelle que le joueur ouvre, manipule et avec laquelle il interagit de manière tactile.

Ce rapport a pour objectif de déconstruire méticuleusement chaque composante graphique nécessaire à la recréation d'une telle expérience. Nous n'analyserons pas seulement le "quoi", mais surtout le "comment" technique : les mathématiques derrière les courbes de Bézier des flèches de ciblage, les shaders HLSL (High-Level Shader Language) responsables des effets de cartes dorées, les algorithmes de génération de maillage procédural, et les pipelines de rendu Unity permettant de gérer la superposition complexe de caméras 2D et 3D.

Pour refaire tout le design avec une fidélité professionnelle, il est impératif de comprendre que le moteur de jeu ne doit pas seulement afficher des images, mais simuler un monde. Chaque carte possède une masse, chaque interaction une inertie, et chaque transition visuelle une courbe d'animation mathématiquement définie pour maximiser la satisfaction utilisateur (le "Juice"). Ce document servira de manuel technique complet pour l'architecte graphique souhaitant implémenter ces systèmes.

2. Architecture du Moteur et Gestion de la Scène (Unity)

La première étape critique dans la reconstruction d'un design de type *Hearthstone* réside dans l'architecture fondamentale de la scène dans le moteur Unity. L'erreur commune est de

penser le jeu comme une simple application 2D. En réalité, il s'agit d'une scène 3D complexe visualisée à travers des techniques de caméras hybrides.[3]

2.1. La Stratégie de l'Empilement de Caméras (Camera Stacking)

Pour obtenir la profondeur visuelle et la clarté de l'interface, il est nécessaire de séparer le rendu en couches distinctes, gérées par des caméras spécifiques. Une caméra unique ne peut pas gérer efficacement à la fois la perspective dramatique du plateau de jeu et la précision au pixel près requise pour les textes et les menus.

Couche (Layer)	Type de Caméra	Rôle Technique	Paramètres Clés
Board Layer (Fond)	Perspective	Rendu du plateau 3D, des décors et des ombres.	FOV : 40-60°. Positionnée en angle pour simuler le regard du joueur assis.
Gameplay Layer (Milieu)	Perspective / Orthographique	Rendu des serviteurs, des sorts et des effets (VFX).	Doit correspondre à la perspective du Board pour que les pieds des serviteurs "touchent" le sol.
Card Hand Layer (Avant-plan)	Perspective Spéciale	Gestion des cartes en main.	Depth élevé. Utilise une matrice de projection modifiée pour éviter que les cartes ne traversent le plateau.
UI Overlay (Interface)	Orthographique	HUD, Mana, Historique, Menus.	Taille fixe. Pas de distorsion de perspective pour garantir la lisibilité du texte.

L'implémentation de cette architecture nécessite l'utilisation des "Layers" et "Culling Masks" de Unity. La caméra du plateau ne doit rendre que les objets marqués sur le layer "Board",

tandis que la caméra UI ne rend que le layer "UI". Le "Depth" (profondeur) des caméras assure l'ordre de rendu final : Board (-1) -> Gameplay (0) -> UI (1).

2.2. La Conversion des Coordonnées (Screen-to-World)

Un défi majeur de cette architecture hybride est l'interaction. La souris du joueur évolue sur un plan 2D (l'écran), mais doit interagir avec des objets 3D (les cartes, le plateau). La simple utilisation de coordonnées d'écran ne suffit pas.
L'implémentation repose sur le "Raycasting" physique. À chaque frame, le moteur doit projeter un rayon depuis la caméra principale à travers la position de la souris :
`Ray ray = Camera.main.ScreenPointToRay(Input.mousePosition);`
Ce rayon intercepte les "Colliders" invisibles placés sur le plateau. Pour gérer le déplacement des cartes (Drag and Drop), on ne fixe pas la carte à la position de la souris. On calcule le point d'intersection du rayon avec un plan mathématique invisible (un Plane géométrique) situé à la hauteur souhaitée au-dessus du plateau. Cela permet à la carte de projeter une ombre réaliste et de donner l'impression de "flotter" physiquement au-dessus de la table, plutôt que d'être collée à l'écran de verre.3

2.3. L'Éclairage et la "Physicalité" des Matériaux

Pour que le design paraisse tangible, l'éclairage ne peut pas être plat. Le plateau utilise des "Lightmaps" (cartes de lumière pré-calculées) pour les ombres statiques de haute qualité, économisant ainsi les ressources GPU. Cependant, les éléments interactifs (cartes, jetons) nécessitent un éclairage dynamique.
L'utilisation de "Matcaps" (Material Capture) ou de shaders standards modifiés est recommandée pour simuler des matériaux comme le bois verni, la pierre ou le métal doré des bordures. Un shader standard Unity (PBR - Physically Based Rendering) peut être trop coûteux pour des centaines de cartes sur mobile ; une approche stylisée avec des textures "Albedo" peintes à la main contenant déjà des informations d'éclairage (baked lighting) est souvent préférée pour l'esthétique "Blizzard", complétée par des "Rim Lights" (lumière de contour) dynamiques pour faire ressortir les silhouettes.2

3. La Flèche de Ciblage : Mathématiques et Génération Procédurale

L'un des éléments les plus emblématiques et complexes à reproduire est la flèche de ciblage (Targeting Arrow). Contrairement à une simple ligne droite, elle se courbe élégamment, contourne les obstacles potentiels (comme la main du joueur) et possède une épaisseur variable et une texture animée. Sa réalisation ne repose pas sur une animation pré-enregistrée, mais sur la génération de maillage en temps réel.[6]

3.1. Théorie Mathématique : Les Courbes de Bézier Cubiques

La forme de la flèche est définie par une **Courbe de Bézier Cubique**. Ce type de courbe nécessite quatre points de contrôle :

1. **P_0 (Source)** : La position de l'objet initiateur (ex: le héros ou la carte jouée).
2. **P_1 (Contrôle 1)** : Un point invisible qui tire la courbe vers le haut et l'avant, créant l'arc initial.
3. **P_2 (Contrôle 2)** : Un point invisible proche de la cible, déterminant l'angle d'arrivée.
4. **P_3 (Cible)** : La position actuelle de la souris ou de la cible survolée.

L'équation vectorielle pour trouver un point $B(t)$ sur la courbe, où t varie de 0 (départ) à 1 (arrivée), est la suivante 6 :

$$B(t) = (1-t)^3 P_0 + 3(1-t)^2 t P_1 + 3(1-t) t^2 P_2 + t^3 P_3$$

Logique d'Implémentation :

Pour dessiner la flèche, vous ne tracez pas une ligne continue. Vous divisez la courbe en segments (par exemple, 30 à 50 segments). Dans une boucle for, vous calculez la position pour t = 0, t = 0.02, t = 0.04, etc., jusqu'à t = 1.

La position des points de contrôle P_1 et P_2 doit être dynamique.

- Si la distance entre P_0 et P_3 est faible, P_1 et P_2 doivent être bas pour éviter une boucle étrange.
- Si la distance augmente, P_1 doit monter sur l'axe Y pour créer un arc dramatique qui passe au-dessus des autres cartes.
- Une formule typique pour P_1 serait : $P_1 = P_0 + \text{Vector3.up} \times (\text{distance} \times 0.5) + \text{Forward} \times (\text{distance} \times 0.3)$.

3.2. Génération de Maillage (Mesh Generation)

Une simple ligne (LineRenderer) peut suffire pour des prototypes, mais pour un design de haute qualité, il faut construire un maillage procédural (un "ruban" 3D).

À chaque pas t le long de la courbe, il faut générer deux sommets (vertices) perpendiculaires à la direction de la courbe pour donner de la largeur à la flèche.7

1. Calcul de la Tangente : Pour orienter correctement les sommets, il faut connaître la direction de la courbe à chaque point. C'est la dérivée première de la fonction de Bézier.

   $$B'(t) = 3(1-t)^2(P_1 - P_0) + 6(1-t)t(P_2 - P_1) + 3t^2(P_3 - P_2)$$

   Le vecteur normalisé de ce résultat donne la direction "avant" ($Forward$) locale de la courbe.
2. Calcul de la Normale : Le vecteur "Droit" ($Right$), perpendiculaire à la courbe (qui définit la largeur), est obtenu par le produit vectoriel (Cross Product) de la tangente et d'un vecteur "Haut" arbitraire (souvent celui de la caméra pour un effet billboard, ou Vector3.up pour un ruban plat).

   $$Right = \text{Vector3.Cross}(Forward, \text{Camera.forward}).\text{normalized}$$
3. **Positionnement des Vertices :**

- $Vertex_{Left} = Point(t) - Right \times \frac{Largeur}{2}$
- $Vertex_{Right} = Point(t) + Right \times \frac{Largeur}{2}$

La largeur peut (et doit) varier en fonction de t. Elle commence fine à la source ($t=0$), s'élargit au milieu, et se connecte à la pointe de la flèche à la fin.

3.3. Animation de Texture et Shader "Flow"

L'aspect magique de la flèche ne vient pas de son mouvement géométrique, mais du mouvement de sa texture. C'est un shader de défilement d'UV (Scrolling UVs).
Le maillage généré possède des coordonnées UV. L'axe U (horizontal) va de 0 à 1 (gauche à droite du ruban). L'axe V (vertical) va de 0 à la longueur de la flèche.
Dans le Shader Graph ou en code HLSL, on ajoute le temps (_Time.y) aux coordonnées UV.
float2 animatedUV = input.uv;
animatedUV.y -= _Time.y * _Speed;
float4 color = tex2D(_MainTex, animatedUV);
Cela crée l'illusion que de l'énergie coule le long de la flèche, même si la géométrie est statique.

3.4. Interaction et "Snapping"

Une fonctionnalité cruciale pour le confort utilisateur est le "Snapping" (aimantation). Lorsque la souris survole une cible valide (détectée par Raycast), le point P_3 de la courbe ne doit plus suivre la souris exactement, mais "sauter" (snap) au centre du Collider de la cible.
Cela donne un feedback tactile immédiat : "Le jeu a compris que vous visez cette cible". La pointe de la flèche (un modèle 3D séparé instancié à la fin de la courbe) doit également s'orienter selon la tangente finale de la courbe pour pointer parfaitement vers le cœur de la cible.7

4. Systèmes de Cartes : Rendu, Shaders Dorés et Parallaxe

La carte est l'atome central du jeu. Sa représentation graphique doit être parfaite. Dans *Hearthstone*, les cartes "Dorées" (Premium) représentent le summum de la technique graphique 2D/3D hybride. Recréer cet effet demande une maîtrise des shaders avancés.[9]

4.1. Structure du Shader "Golden Card"

L'effet "doré" n'est pas une vidéo (ce qui serait trop lourd en mémoire pour des centaines de cartes). C'est une composition de plusieurs couches visuelles pilotées par des mathématiques en temps réel.

Architecture du Shader (HLSL/Shader Graph) :

1. **Texture de Base (Albedo) :** L'illustration normale.
2. **Masques (Channels Packing) :** Une texture technique où chaque canal de couleur

(Rouge, Vert, Bleu, Alpha) contient un masque noir et blanc différent.
    - *Canal R :* Masque de "Foil" (zones brillantes/dorées).
    - *Canal G :* Masque de distorsion (zones qui doivent bouger, comme des cheveux ou du feu).
    - *Canal B :* Masque de particules ou de brillance secondaire.
3. **Texture de Bruit (Noise/Flow Map) :** Une texture de nuages ou de bruit de Perlin qui défile en continu.

Logique de Distorsion :
Pour animer une image statique (faire onduler une cape), le shader utilise le déplacement UV. On échantillonne la texture de bruit, et on utilise sa valeur pour décaler les coordonnées de lecture de la texture principale.
float2 distortion = tex2D(_NoiseTex, input.uv + _Time.x * _Speed).rg;
float2 finalUV = input.uv + (distortion * _DistortionIntensity * input.mask.g);
Cette technique permet de donner vie à des éléments statiques sans rigging 3D coûteux.12

4.2. Effet de Parallaxe (Profondeur)

L'effet le plus impressionnant est la parallaxe : l'illustration semble avoir de la profondeur à l'intérieur du cadre de la carte. Le fond bouge différemment du premier plan lorsque l'on bouge la souris ou le téléphone (gyroscope).
Techniquement, cela est réalisé en décalant les UVs en fonction de l'angle de vue (View Direction).10

1. Calculer le vecteur de vue : float3 viewDir = normalize(_WorldSpaceCameraPos - input.worldPos);
2. Calculer le décalage : float2 parallaxOffset = viewDir.xy * _ParallaxScale;
3. Appliquer ce décalage de manière inverse aux couches de fond et de premier plan. Si le fond se déplace vers la gauche quand on regarde à droite, il semble plus "loin".

4.3. Simulation de Réflection (Foil/Brillance)

L'aspect "doré" brillant utilise une technique de Reflection Mapping ou MatCap. Le shader ne calcule pas de vrais reflets de lumière (trop coûteux). À la place, il projette une texture d'image de "reflet doré" sur la carte en fonction de la normale de la surface et de la vue caméra.
Lorsque la carte tourne, la texture de reflet glisse sur la surface, simulant une réaction métallique à la lumière. L'ajout d'une texture de "poussière d'étoiles" (glitter) en mode additif, masquée par le Canal R, crée les scintillements caractéristiques des cartes rares.13

5. Interactions Physiques et le "Juice" (Ressenti de Jeu)

Un design graphique réussi ne se limite pas à ce qui est affiché, mais inclut la manière dont les objets réagissent. Le concept de "Juice" (jus) réfère à la quantité de feedback audiovisuel

pour chaque interaction.

5.1. Physique du "Drag and Drop" (Glisser-Déposer)

Il est crucial de ne jamais lier directement la position (Transform.position) d'une carte à la position de la souris lors d'un glisser-déposer. Cela crée un mouvement rigide et artificiel.
Il existe deux méthodes supérieures pour simuler le poids et l'inertie 15 :

1. Interpolation (Lerp) avec Retard :
   Dans la boucle Update, on déplace la carte vers la souris avec une vitesse finie.
   transform.position = Vector3.Lerp(transform.position, targetMousePos, Time.deltaTime * smoothFactor);
   Cela crée un léger retard ("lag") qui donne une impression de masse.
2. Joints Physiques (Spring Joint) :
   C'est la méthode la plus réaliste. On attache un composant SpringJoint au Rigidbody de la carte. La souris contrôle un objet "Ancre" invisible (Kinematic Rigidbody). Le joint tire la carte vers l'ancre. Cela permet à la carte d'avoir des oscillations naturelles, de dépasser légèrement la cible (overshoot) et de réagir aux collisions avec d'autres objets du décor.

5.2. Rotation Dynamique (Banking)

Une carte déplacée latéralement doit s'incliner pour simuler la résistance de l'air. C'est un calcul simple basé sur la vélocité.
float tiltX = (currentVelocity.x) * tiltSensitivity;
Quaternion targetRotation = Quaternion.Euler(tiltX, 0, -tiltX);
Plus le joueur bouge la souris vite, plus la carte penche. C'est un détail subtil mais essentiel pour le dynamisme.17

5.3. Principes d'Animation : Squash and Stretch

Inspiré de l'animation traditionnelle Disney, chaque interaction UI doit utiliser le "Squash and Stretch" (Écrasement et Étirement).[18]

- **Anticipation :** Avant de piocher une carte, celle-ci doit reculer légèrement ou se gonfler.
- **Action :** Lors de l'impact sur le plateau, la carte ne doit pas s'arrêter net.
- **Impact :** À l'instant où la carte touche le plateau, son échelle (Scale) doit être modifiée.
  - Frame 1 (Impact) : Scale X = 1.2, Scale Y = 0.8 (La carte s'écrase sous le choc).
  - Frame 5 (Rebond) : Scale X = 0.9, Scale Y = 1.1 (La carte rebondit et s'étire).
  - Frame 10 (Repos) : Scale X = 1.0, Scale Y = 1.0.
    Cette déformation procédurale, gérée par des Coroutines ou des bibliothèques de Tweening (comme DOTween), vend l'idée que la carte est un objet physique flexible et non une image PNG rigide.21

6. Le Système d'Animation et les Courbes (Easing)

Pour recréer le design d'animation fluide de *Hearthstone*, il faut bannir l'interpolation linéaire

(Linear Easing). Rien dans la nature ne bouge à vitesse constante du début à la fin.

6.1. Mathématiques des Courbes d'Assouplissement (Easing Functions)

Chaque mouvement dans le jeu (pioche, pose, attaque) utilise une courbe spécifique qui dicte la vitesse en fonction du temps.[22]

Type de Courbe	Usage dans le Design	Caractéristique Mathématique
Ease-Out Back	Piocher une carte, faire apparaître une fenêtre.	La carte dépasse légèrement sa destination finale avant de revenir. Formule incluant un dépassement $c1$.
Ease-In Cubic	Projectiles, boules de feu.	Démarrage lent, accélération violente à la fin. Donne une impression de puissance à l'impact.
Elastic / Bounce	Erreurs, chocs, atterrissages lourds.	Oscillation sinusoïdale amortie. Simule un ressort ou une balle en caoutchouc.
Ease-In-Out Sine	Mouvements de caméra, transitions douces.	Accélération douce, décélération douce. Naturel et non agressif.

6.2. Implémentation Technique

Dans Unity, l'utilisation de l'AnimationCurve dans l'inspecteur permet aux designers de dessiner visuellement ces courbes. Le code évalue ensuite cette courbe :
float t = timer / duration;
float curvedT = myAnimationCurve.Evaluate(t);
transform.position = Vector3.Lerp(start, end, curvedT);
L'utilisation de Evaluate permet de modifier le "feeling" du jeu sans réécrire une seule ligne de code, simplement en ajustant les tangentes de la courbe dans l'éditeur.

7. Gestion des Effets Visuels (VFX) et Particules

Les VFX sont le langage visuel du jeu. Ils communiquent les règles (dégâts, soins, buffs) par la couleur et le mouvement.

7.1. Architecture des Sorts et Projectiles

Un effet visuel comme une "Boule de Feu" est un composite complexe, pas un simple sprite [5] :

- **The Head (Tête) :** Le projectile principal. Souvent un mesh 3D stylisé avec un shader additif brillant.
- **The Trail (Traînée) :** Pour les sorts magiques, on utilise un TrailRenderer. Pour donner un aspect "peint", la texture de la traînée doit être un gradient qui s'érode (Dissolve) avec le temps. Pour les projectiles physiques (flèches, couteaux), on préfère des systèmes de particules émettant des sprites de fumée ou de poussière.[24]
- **The Glow (Lueur) :** Un sprite billboard (toujours face caméra) très grand et très doux qui donne l'impression d'illumination sans utiliser de vraie lumière coûteuse.
- **The Distortion (Distortion) :** Un système de particules utilisant un shader de réfraction (GrabPass). Il ne dessine pas de couleur, mais déforme les pixels derrière lui, simulant la chaleur ou l'onde de choc.

7.2. L'Art de l'Impact

L'impact est plus important que le projectile. Il doit être synchronisé avec :

1. **Le Flash Blanc :** Le shader du serviteur touché passe brièvement en blanc pur (Emission à 100%).
2. **Le Shake (Secousse) :** La caméra reçoit une impulsion de translation aléatoire qui s'amortit rapidement.
3. **Les Particules Directionnelles :** Les débris doivent voler dans la direction opposée à l'impact (conservation de la quantité de mouvement).

7.3. Optimisation des VFX (Overdraw)

Une erreur critique à éviter est l'"Overdraw" (dessiner plusieurs fois le même pixel). Les effets transparents (particules de fumée) empilés tuent les performances mobiles.
Solution : Utiliser des meshes opaques ou "Cutout" autant que possible pour le cœur des effets, et garder la transparence pour les bords uniquement. Limiter le nombre maximum de particules et utiliser le "Texture Sheet Animation" pour simuler du mouvement complexe avec un seul quad.24

8. La Mécanique "Discover" et le Flou d'Interface

Le mécanisme "Discover" (Découverte), où le joueur choisit 1 carte parmi 3, est un cas d'école de design UX focalisé.[26] L'élément clé ici est la focalisation de l'attention par le flou de

l'arrière-plan.

8.1. Implémentation du Flou en Temps Réel

Flouter toute la scène 3D en temps réel est extrêmement coûteux (Shader Gaussian Blur). *Hearthstone* utilise une astuce d'optimisation intelligente [27] :

1. **Capture (Snapshot) :** Au moment où l'effet Discover se déclenche, le jeu capture une seule frame de la caméra de jeu dans une RenderTexture.
2. **Downsampling :** Cette texture est immédiatement réduite (par exemple, divisée par 4 ou 8 en résolution). La réduction de résolution crée un flou naturel "pixelisé" et réduit drastiquement le coût de traitement.
3. **Application du Shader :** Un shader de flou léger est appliqué sur cette petite texture.
4. Affichage Statique : Le rendu de la scène 3D (le plateau) est désactivé ou caché par un panneau UI (RawImage) qui affiche cette texture floue statique.
   Cela donne l'illusion d'un flou de profondeur de champ dynamique, mais c'est en réalité une image fixe. Cela libère toute la puissance du GPU pour afficher les 3 cartes du choix avec la plus haute qualité possible (particules, shaders dorés, lumières).

8.2. Mise en Page et Probabilités

L'interface de Discover présente trois cartes alignées horizontalement. Techniquement, ce sont des instances complètes de l'objet carte, mais avec des scripts d'interaction désactivés (on ne peut pas les attaquer, juste les cliquer).
L'algorithme de génération des choix doit être reflété dans le code : le système tire 3 IDs de cartes depuis la base de données. Il existe un "poids" (weighting) spécifique : les cartes de la classe du joueur ont 4x plus de chances d'apparaître que les cartes neutres.26 Le design doit gérer les cas limites : que se passe-t-il si le texte de la carte est trop long? Le système de rendu de texte dynamique (TextMeshPro dans Unity) doit ajuster la taille de la police automatiquement pour s'adapter à la zone de description.

9. L'Expérience d'Ouverture de Paquets (Pack Opening)

C'est le moment le plus "tangible" du jeu. Il combine physique, anticipation et récompense visuelle.[28]

9.1. Simulation Physique du Paquet

Le paquet de cartes n'est pas une icône, c'est un objet 3D avec un Rigidbody.

- **Dragging :** Quand le joueur traîne le paquet vers la zone d'ouverture, il pendouille comme un pendule (Hinge Joint).
- **Lâcher :** Lorsqu'on le relâche, la gravité prend le relais. Il tombe dans le réceptacle.
- **Explosion :** L'ouverture utilise la fonction Rigidbody.AddExplosionForce. Les 5 cartes sont instanciées au même point (dans le paquet) et une force explosive est appliquée

pour les éjecter dans toutes les directions. C'est une simulation physique réelle, pas une animation pré-calculée, ce qui rend chaque ouverture légèrement unique.
- **Ordonnancement :** Après l'explosion, un script reprend le contrôle (Kinematic) et utilise des Tweens pour ranger proprement les cartes face cachée en arc de cercle.

9.2. Le Système de Rareté (Glow)

Avant de retourner la carte, le joueur peut survoler le dos. Un shader de "Hover" détecte la rareté de la carte (Common, Rare, Epic, Legendary).

- **Common :** Pas de lueur (ou blanche très faible).
- **Rare :** Lueur Bleue.
- **Epic :** Lueur Violette.
- Legendary : Lueur Orange intense avec particules pulsantes.
  Ce "Teasing" visuel augmente l'anticipation. La lueur est gérée par une "Emission Color" sur le matériau du dos de carte, couplée à un système de particules en boucle locale autour de la carte.28

10. Environnement et Éléments Interactifs ("Clickables")

Le plateau de jeu possède quatre coins interactifs. Ce ne sont pas de simples décors, mais des mini-jeux physiques.[2]

10.1. Structure des Clickables

Chaque coin est un "Prefab" complexe contenant :

- **Modèles 3D :** Bâtiments, plantes, engins.
- **Colliders Invisibles :** Des MeshCollider simplifiés qui interceptent les clics.
- **Contrôleurs d'Animation :** Des machines à états (State Machines).
  - *Idle :* Animation boucle légère (fumée qui sort d'une cheminée).
  - *Trigger :* Animation déclenchée par le clic (catapulte qui tire, vitre qui se brise).
  - *Reset :* Logique pour remettre l'objet en place ou le laisser cassé.
- **Sons Spatialisés :** Le son doit venir de la position 3D du coin (AudioSource 3D), renforçant l'immersion.

10.2. Éclairage et Texture Projection

Pour que les objets semblent "posés" sur le plateau et non flottants, l'éclairage est crucial. Le sol du plateau est souvent une texture unique peinte à la main en haute résolution (4K ou 2K). Pour lier les objets au sol, on utilise l'Ambient Occlusion (AO). Dans Unity, cela peut être "bakes" (cuit) directement dans la texture du sol sous les objets statiques. Pour les objets mobiles, on utilise des "Blob Shadows" (projecteurs d'ombres simplifiés) qui sont des textures noires floues avec transparence, placées juste au-dessus du sol sous l'objet, ce qui est

beaucoup moins coûteux que les ombres en temps réel pour le décor.2

11. Conclusion et Recommandations d'Implémentation

La recréation du design graphique d'un jeu comme *Hearthstone* est un exercice de synthèse. Ce n'est pas seulement du dessin (Art 2D), ce n'est pas seulement du code (C#), et ce n'est pas seulement de la 3D. C'est l'intégration étroite de ces trois disciplines.

Feuille de Route pour le Développement :

1. **Phase Architecture :** Mettre en place le système de couches de caméras (Camera Stacking) et le gestionnaire d'input (Raycasting) dès le jour 1. C'est la fondation.
2. **Phase "Feel" :** Implémenter le système de déplacement de cartes avec inertie (Spring Joint/Lerp) et les courbes d'animation (Easing). Tant que déplacer un cube gris n'est pas "satisfaisant", ne pas passer à l'étape suivante.
3. **Phase Visuelle (Shaders) :** Développer le shader de carte maître (Master Card Shader) gérant la parallaxe et les effets dorés. C'est l'asset le plus vu du jeu, il doit être parfait.
4. **Phase "Juice" :** Ajouter les particules, les secousses de caméra et les animations d'impact.

En suivant cette approche technique rigoureuse, en respectant les mathématiques des courbes et la physique des interactions, il est possible de reproduire la qualité tactile et immersive qui définit les standards actuels du genre. Le secret ne réside pas dans la complexité d'un seul élément, mais dans l'harmonie et la réactivité de l'ensemble du système.

Sources des citations

1. Video: Blizzard's art design principles for Hearthstone, consulté le janvier 5, 2026, https://www.gamedeveloper.com/design/video-blizzard-s-art-design-principles-for-i-hearthstone-i-
2. Design and development of Hearthstone, consulté le janvier 5, 2026, https://hearthstone.fandom.com/wiki/Design_and_development_of_Hearthstone
3. How Hearthstone looks 3D inside the Game Engine - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/hearthstone/comments/65cs6o/how_hearthstone_looks_3d_inside_the_game_engine/
4. How would I do a cardhand like in hearthstone (drag and drop) but still part of the GUI (follow camera,etc)? : r/godot - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/godot/comments/826eld/how_would_i_do_a_cardhand_like_in_hearthstone/
5. Developer Insights: The Art Behind THE SCIENCE - Hearthstone, consulté le janvier 5, 2026, https://hearthstone.blizzard.com/en-us/blog/22552047/developer-insights-the-art-behind-the-science
6. How to work with Bezier Curve in Games with Unity - Game Developer, consulté le janvier 5, 2026,

https://www.gamedeveloper.com/business/how-to-work-with-bezier-curve-in-games-with-unity
7. Game Dev Tutorial: How to recreate the targeting arrow in Slay the Spires - YouTube, consulté le janvier 5, 2026, https://www.youtube.com/watch?v=FpuH303FYYU
8. Hearthstone Like Unit Move Orders Arrow Graphics - Free Tutorial - Construct 3, consulté le janvier 5, 2026, https://www.construct.net/en/tutorials/hearthstone-unit-move-orders-532
9. Shaders Case Study - Hearthstone Golden Cards : r/gamedev - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/gamedev/comments/4odmik/shaders_case_study_hearthstone_golden_cards/
10. Parallax Magic Cards - Stoyan Dimitrov - WordPress.com, consulté le janvier 5, 2026, https://stoyan3d.wordpress.com/2021/07/30/parallax-magic-cards/
11. Hearthstone custom gold cards - Real Time VFX, consulté le janvier 5, 2026, https://realtimevfx.com/t/hearthstone-custom-gold-cards/1792
12. Shaders Case Study - Hearthstone Golden Cards : r/Unity3D - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/Unity3D/comments/4odk8y/shaders_case_study_hearthstone_golden_cards/
13. Implementing Hearthstone-ish Card Effects : r/Unity3D - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/Unity3D/comments/hxlymi/implementing_hearthstoneish_card_effects/
14. Hearthstone Golden Cards Shader : r/Unity3D - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/Unity3D/comments/4nhwjg/hearthstone_golden_cards_shader/
15. How to drag object without elasticity with joint - Stack Overflow, consulté le janvier 5, 2026, https://stackoverflow.com/questions/53702084/how-to-drag-object-without-elasticity-with-joint
16. How to make card movement behave like those in Hearthstone and Eternal CCG?, consulté le janvier 5, 2026, https://gamedev.stackexchange.com/questions/137265/how-to-make-card-movement-behave-like-those-in-hearthstone-and-eternal-ccg
17. Hearthstone - Parallax Effect in Unity - YouTube, consulté le janvier 5, 2026, https://www.youtube.com/watch?v=1_pBJK2vNQw
18. Mastering the Squash & Stretch Principle in Animation - CGWire Blog, consulté le janvier 5, 2026, https://blog.cg-wire.com/squash-stretch-principle/
19. SQUASH & STRETCH - The 12 Principles of Animation in Games - YouTube, consulté le janvier 5, 2026, https://www.youtube.com/watch?v=1kFRU_xBZnE
20. 1. Squash & Stretch - 12 Principles of Animation - YouTube, consulté le janvier 5, 2026, https://www.youtube.com/watch?v=haa7n3UGyDc
21. Principles of Animation- Squash & Stretch : r/gamedev - Reddit, consulté le janvier 5, 2026,

https://www.reddit.com/r/gamedev/comments/fbpjdd/principles_of_animation_squash_stretch/
22. Easing Functions in the Animation Process - Marionette Studio, consulté le janvier 5, 2026, https://marionettestudio.com/easing-functions-in-the-animation-process/
23. Easing Functions Cheat Sheet, consulté le janvier 5, 2026, https://easings.net/
24. r/Unity3D on Reddit: Added a "persistent particle trails" performance option for magic. ON - Trails finish simulating and smoothly dissipate after projectiles make impact. OFF, consulté le janvier 5, 2026, https://www.reddit.com/r/Unity3D/comments/1bngcf5/added_a_persistent_particle_trails_performance/
25. Trail & Impact VFX Production - 80 Level, consulté le janvier 5, 2026, https://80.lv/articles/trail-impact-vfx-production
26. Hearthstone: Creating the Discover Mechanic - IGN, consulté le janvier 5, 2026, https://www.ign.com/articles/2016/01/17/hearthstone-creating-the-discover-mechanic
27. Simple Blur and Pixilation filter Shader - Unity, Godot Tutorial - YouTube, consulté le janvier 5, 2026, https://www.youtube.com/watch?v=yQzWt5FT930
28. Pack Opening Animations Explanation : r/masterduel - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/masterduel/comments/1ens9gk/pack_opening_animations_explanation/
29. Open Packs - New Hearthstone Wiki, consulté le janvier 5, 2026, https://hearthstone.wiki.gg/wiki/Open_Packs
30. Pack opening guide visualized : r/hearthstone - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/hearthstone/comments/17urxx1/pack_opening_guide_visualized/
31. Working on some Hearthstone-style board interactions for my card game—just added water, grass & dust VFX. Strategy for the brain, fidgeting for the fingers. : r/Unity3D - Reddit, consulté le janvier 5, 2026, https://www.reddit.com/r/Unity3D/comments/1lf2fi7/working_on_some_hearthstonestyle_board/