INTRODUCTION À

UNITY

GAME ENGINE

- Game Loop
- Moteur Graphique / physique
- Outils pour faciliter la programmation
- Nombreux Game Engine sur le marché
  - Unity
  - Unreal Engine
  - CryEngine / Lumberyard
  - GameMaker
  - ...

UNITY

- Lancé en 2005 uniquement sur iOS
- Objectif: "Démocratiser le dévelopement de jeux vidéo"
- Premier game engine complet gratuit
- Aujourd'hui: 27 plateformes supportées
- 35% de parts de marché dans le jeu mobile
- D'autres industries que le jeu vidéo utilisent Unity
  - Cinéma
  - Architecture
  - Ingénierie
  - ...

TEXTE

ARCHITECTURE D'UN JEU UNITY

- Un jeu est composé de "Scene"
- Une scène est composée de "GameObject"
- Un GameObject contient des "Component"
- Les Component déterminent la fonction du GameObject
- Unity fournit un ensemble de Component tout faits
  - Transform
  - Collider
  - Camera
  - ...
- Un type de Component permet de rajouter notre propre logique: les "Scripts"

L'EDITEUR DE UNITY

- Scene: Pour positionner les GameObject dans l'espace
- Game: Pour tester le jeu
- Hierarchy: Tous les objets présents sur la scène
- Project: Tous les fichiers permettant de faire un build du jeu
- Inspector: Les propriétés du GameObject sélectionné
- Console: Log les erreurs / warning / debug

SCRIPTS

- Héritent de la classe "MonoBehaviour"
- Les propriétés publiques sont visibles dans l'Inspector
- Certaines méthodes sont appelées automatiquement par la Game Loop (Event Functions)
  - Start
  - Update
  - OnCollisionEnter
  - ....

QUELQUES COMPONENTS IMPORTANTS

- Transform / RectTransform
- RigidBody / RigidBody2D
- Collider / Collider2D
- Mesh / MeshFilter
- Camera
- Light

VECTOR3, VECTOR2, QUATERNION

- Classes très importantes dans Unity
- Servent à représenter toutes sortes de choses:
  - Position
  - Direction
  - Distance
  - Echelle
  - Rotation
- Cas particulier: la rotation
  - Représentée dans l'inspecteur comme un Vector3 d'angles d'Euler
  - Stocké comme "Quaternion" = rotation de w degrés autour d'un axe x, y, z
  - Il existe des méthodes de conversion
    - Quaternion myRotation = Quaternion.Euler(myVector3);
    - Vector3 myVector3 = myRotation.EulerAngles();

TRANSFORM

- Component présent dans tout GameObject
  - transform.position : position de l'objet
  - transform.rotation : rotation de l'objet
  - transform.scale : échelle de l'objet
- Pour modifier la position d'un objet sur la scène on peut modifier les valeurs du transform
  - transform.Translate(Vector3 offset);
- Cas particulier: RectTransform pour les éléments de UI

RÉFÉRENCER UN AUTRE OBJET

- Méthode "barbare"
  - GameObject.Find("MonObjet")
- Moins violent
  - GameObject.FindWithTag("Car")
  - GameObject.FindObjectWithType<Car>()
- Stocker la référence dans une propriété
  - public GameObject myObjectReference

RIGIDBODY

- Composant permettant à un GameObject d'être affecté par la physique
- On distingue RigidBody (pour la physique 3D) et RigidBody2D
- Un RigidBody peut etre
  - Dynamique (100% affecté par la physique)
  - Cinématique (on peut lui donner du mouvement, mais pas appliquer une force)
  - Statique (il ne peut pas bouger)
- On peut faire beaucoup de chose avec un Rigidbody
  - Lui donner une vitesse
  - Lui appliquer une force
  - Réagir à une collision

RIGIDBODY-EXEMPLES

- Donner une vitesse à un RigidBody
  - rigidBody.velocity = new Vector3 (x, y, z);
- Appliquer une force à un RigidBody
  - rigidBody.AddForce(new Vector3(x, y, z));
- Attention:
  - RigidBody2D pour la 2D

COLLIDER

- Un collider détermine la zone de collision d'un GameObject
- Il en existe de multiples formes
  - BoxCollider
  - SphereCollider
  - CircleCollider2D
- Un Collider peut être défini comme "trigger"
  - Il ne produit pas de collision physique mais on peut détecter si un autre collider pénètre dedans.
  - Utile pour implémenter des zones de détection (ex: zone d'aggro)

COLLIDER – EVENT FUNCTIONS

- Détecter une collision
  - void OnCollisionEnter(Collision col)
  - void OnCollisionEnter2D(Collision2D col)
- Détecter un trigger
  - void OnTriggerEnter(Collider col) <---- !!!Collider!!!
  - void OnTriggerEnter2D(Collider2D col)
- Autres fonctions
  - OnTriggerExit
  - OnTriggerStay
  - ....

MATRICE DE COLLISION

- On peut configurer le moteur physique
  - Edit / Project Settings / Physics (ou Physics2D)
- Entre autres choses il y a la matrice de collision
- Cette matrice définit quelles couches (layer) peuvent entrer en collision
- On peut définir pour chaque GameObject à quelle couche il appartient

COLLIDERS, TRIGGERS ET RIGIDBODY

- Pour qu'un trigger ou une collision soit détectée
  - Les deux GameObjects doivent avoir des Collider
  - Les colliders doivent être dans la même dimension (2D avec 2D, 3D avec 3D)
  - Au moins un des objets doit avoir un Rigidbody
  - La collision doit être autorisée dans la matrice de collision
  - Les deux types de collider (Trigger, Cinématique, Dynamique, ...) doivent pouvoir accepter la collision / trigger
    - cfr documentation: https://docs.unity3d.com/Manual/CollidersOverview.html

RÉFÉRENCER UN COMPOSANT

- Récupérer le composant d'un objet
  - gameObjectRef.GetComponent<Collider>()
- Stocker un composant dans une propriété / variable
  - public Collider colliderRef

JOUER AVEC LE TEMPS

- Temps de calcul de la dernière frame
  - Time.deltaTime
  - Time.fixedDeltaTime
- Ralentir / arrêter le temps
  - Time.timeScale
- Autres
  - Time.realTimeSinceStartup
  - Time.time
  - Time.frameCount
  - ...

INSTANCIER DES OBJETS

- Prefab
  - GameObject "sauvé" sous forme de fichier
  - Un préfab est toujours instancié de manière identique
  - Après, s'il est modifié il n'impacte pas les autres instances
- Instanciation
  - Instantiate(myPrefab, myPosition, myRotation)
  - Notez qu'en anglais, "instantiate" prend un T et pas un C