Alexandre Lattavo

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

Retour sur le C# et Unity
Préparation au workshop Tower Defense

- Cours
- Exercice guidé
- Exercice seul

Retour sur Unity et le C#

On retourne sur des notions vues en prépa et on en ajoute quelques unes

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

On passe sur tous les concepts que vous avez besoin de connaître :

- POO
- Mémoire
- Utilisation de Unity pour créer des éléments de jeu : MonoBehaviour, déplacement, instantiations, etc...

Construire les éléments d'un TD

- Renforcement des notions de programmation
- Utilisations plus avancée de Unity pour compléter les fonctionnalités d'un TD

Finaliser un prototype

- Scenes
- User Interface

- Unity-centered : scenes, user interface et build

Si on a le temps :

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Retour sur quelques definitions :
  - hardware / drivers / software
  - Instructions / operators / programme
  - Logiciel / Moteur de jeu
- Multi plateforme sur Unity
  - Assembly decompiling
  - Intermediate Language
  - IL2CPP
- Microsoft / .Net / C#
  - Documentation / Guideline

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Retour sur les fondamentaux de la programmation
  - Primitive type : int / float, string
  - Branching
  - Struct / Class

- La programmation comme industrie : l'importance de suivre les standards

- KISS : Keep it Simple, Stupid
- Single Responsability
- Don't Repeat Yourself : un code dupliqué => factoriser
- Law of Demeter : parle à ton voisin qui parle au sien
- Clarté du code avant la concision : le but est de lire et modifier facilement un code, pas de le rendre le plus concis possible
- YAGNI (You Aren't Gonna Need It) : ou sous une autre forme, on ne répare pas ce qui n'est pas cassé
- Open for extension, closed for modification : tout en private par défaut et on expose en public uniquement le necessaire
- Et plein d'autres

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Principes de bases de la POO
- Différences MonoBehaviour / POCO
- Encapsulation : organiser un problème complexe en plusieurs problèmes simples.

- Exercice guidé
  - Un actor qui se déplace à une position
  - Un actor qui se déplace à une liste de positions
  - Premier design pattern : simple State machine

- Exercice
  - Créer une class Timer qui encapsule le comportement.

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Héritage : Étendre des fonctionnalités
  - POCO : Plain Old CLR Object (CLR : Common Language Runtime)
  - Exploration de Object
  - Object / MonoBehaviour / ScriptableObject

- Exercice guidé :
  - Créer une classe abstraite Weapon et deux dérivées
  - Créer une classe Projectile
  - Créer un Damageable

- Exercice :
  - Ajouter une nouvelle Weapon
  - Modifier la classe Projectile en abstraite AProjectile et créer deux dérivée (une par défaut, une à proposer)

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Polymorphisme :
  - Capacité d'une class à être considérée du type d'une class mère
  - Modifier les comportements d'une classe mère

- Introduction aux casts : (type), is, as

- Mémoire :
  - Heap / Stack
  - Value Type ou Reference Type
  - Primitive type
  - Exemple struct / class
  - Cas particulier : strings

https://www.dofactory.com/net/design-patterns

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Design Patterns (couleur == fréquence d'utilisation en jeu vidéo) :
  - Command ➢ Encapsuler un comportement
  - Singleton ➢ Unique et accessible par tous le monde
  - State machine ➢ Encapsuler des comportements exclusifs en état avec un seul comportement actif
  - Decorator ➢ Ouvrir une fonctionnalité à l'ajout / modification de comportement
  - Component Model
  - Factory ➢ Encapsuler et cacher la complexité d'une creation d'objet
  - Facade ➢ Organiser un ensemble de sous-système
  - Observer ➢ Regrouper la communication inter-object dans par un intermédiaire pour réduire les dépendences

- Exercice guidé :
  - Encapsuler le comportement d'un actor en State et utiliser une State Machine.
  - Update l'UI avec des events

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Notions abordées :
  - Espace Local / World / Screen
  - Rotations : Euler et Quaternion, gimbal lock et accumulation de rotation
  - Linear Interpolation / Spherical Linear Interpolation
  - Layer et bitmask (!!! très différent des tags !!!)
  - Raycast

- Exercice guidé :
  - Ajouter une class Tower qui s'oriente vers le plus proche ennemis
  - Trier les targets possible avec les layers ou les tags
  - Ajouter des tuiles interactable pour y construire des tours

- Exercice : Créer une Tower "support" qui vise les autres tours et leur donne de la vie
  - Trier par layer pour trouver les bonnes target
  - Ajouter un Projectile qui heal le Damageable touché
  - Modifier Damageable pour déclencher un heal

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Design pattern MVC : Model View Controller

- Gestion des scenes :
  - Single / Additive
  - Build settings

- Exercice guidé :
  - Créer une scène bottleneck : EntryPoint
  - Ajouter une scène Main menu : intro UI
  - Créer un menu de choix de niveau : database de scène

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Camera et Canvas settings : space, scaling et RectTransform

- Exercice guidé : Créer un UI ingame basique
  - UIManager en singleton
  - Afficher le nombre de tours
  - Event Binding

- Exercice :
  - Afficher le nombre d'Actor
  - Afficher le nombre d'Actor tué

- Exercice guidé : Grid Layout, HV Layout, Element Layout

- Exercice : Refaire l'interface correctement layouté

- Exercice guidé : Créer une barre de vie en World Space avec des ancres

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Choix de patterns sur Unity :
  - Prefab vs Scriptable Object vs Scene
- Premier pas dans l'optimisation :
  - Find / FindObjectXXX vs Direct Reference
  - Fonctionnement d'une string : char*, implicit cast, new(), operator+, comparison
  - Boxing / Unboxing
- Coding style
- Notions avancées :
  - Interface
  - Operator overloading
  - Binary operator et Boolean algebra
  - Algorithme de tri
- Debugging :
  - Breakpoint
  - Debug mode / step by step / step into

Retour sur Unity et le C#

Fondamentaux de la programmation object

- POO : Encapsulation
- POO : Héritage
- POO : Polymorphisme et mémoire
- Design patterns

Construire les éléments d'un TD

Finaliser un prototype

- Scenes
- User Interface

Si on a le temps :

- Exercice : Encapsuler correctement les comportements adéquats
  - Wave / Wave Database / Wave Manager
  - Actor Factory
  - Tower Factory

- Exercice : Exporter les settings qui vous semble adéquat vers des "databases"
  - Actor settings ? Speed