Document de conception

Classes abstraites

Afin de pouvoir travailler en commun sur le projet, nous avons créé des classes abstraites pour qu'elles soient implémentées dans chaque jeux. Ces classes sont utilisées pour implémenter les différents patterns mais servent également aux classes "non abstraites" communes à tous les jeux. Par exemple les chargements des fichiers XML.

ModuleIHMAbstrait

Ce module gère l'affichage du jeu.

Chaque scénario implémente cette classe pour spécifier l'affichage de son jeu.

ModuleStatsAbstrait

Ce module gère les statistiques du jeu.

Defence Tower

Niveau atteint, ennemis tués.

Echec

Nombre de coups total joués.

ZoneAbstraite

Cette classe représente une position dans le plateau de jeu pouvant contenir un ou plusieurs objets et/ou personnages.

Defence Tower

Champ de bataille, zone du château

Echec

Case du plateau de jeu.

Trafic

Routes.

AccesAbstrait

Un accès permet de faire le lien entre deux zones.

Defence Tower

Fait le lien entre le champ de bataille et le château.

Echec

Fait le lien entre chaque case du plateau.

Trafic

Faire le lien entre deux portions de route.

PersonnageAbstrait

Cette classe représente un personnage du jeu avec des attributs tels que le nombre de points de vie, le nom…

Defence Tower

Définition du château et des ennemis attaquant le château.

Echec

Pions blanc et noir du jeu d'échec.

Trafic

Véhicules sur la route.

ComportementCombattreAbstrait

Un comportement permettant de combattre sera défini pour chaque personnage.

Defence Tower

Comportement de combat des ennemis.

Echec

Comportement de combat des pions.

ComportementSeDeplacerAbstrait

Un comportement gérant le déplacement sera défini pour chaque personnage.

Defence Tower

Déplacement des ennemis vers le château.

Echec

Dépacement des pions sur l'échiquier.

Trafic

Déplacement des voitures sur la route.

EtatAbstrait

Un personnage pourra avoir plusieurs Etats dans lequels ses actions pourront différer.

Defence Tower

État des tirs du château.

Echec

État d'une pièce (ex : vie, mort).

FabriqueAbstraite

Cette classe permet d'instancier des personnages, des objets spécifiques à chaque jeu.

Defence Tower

Instanciation des personnages : Ennemis et Chateau.

Echec

Instanciation des pions.

Trafic

Instanciation des véhicules et des feux.

SujetObserveAbstrait / ObservateurAbstrait

Cette classe représente un sujet qui notifie une liste d'observateurs du changement de son état

Trafic

Feux de signalisation qui indiquent à l'autre feu synchronisé de passer au vert ou au rouge.

ComportementChargerSimulationAbstrait

Cette classe permet de charger les jeux en fonction du jeu choisi. Chaque chargement implémente cette classe et initialise les jeux pour ensuite les lancers.

Tours de jeu

Tous les jeux tournent de la même façon, ils sont tous lancés depuis la fenêtre principale où un thread est lancé pour appeler la fonction TourDeJeu de Simulation qui est commune à tous les jeux.

Simulation

```
using System;
using System.Collections.Generic;
namespace ProjetDesignPattern
{
    public class Simulation
    {
        public String Nom { get; set; }
        public int vitesse=1000;
        public bool finDuJeu = false;
        public List<SujetObserveAbstrait> listeSujetsObserves;
        public List<PersonnageAbstrait> listePersonnages;
        public List<ZoneAbstraite> listeZones;
        public List<ConflitAbstrait> listeConflits;
        public List<ObjetAbstrait> listeObjets;
        public ModuleStatsAbstrait ModuleStats { get; set; }
        public ModuleIHMAbstrait ModuleIHM { get; set; }
        public FabriqueAbstraite fab { get; set; }
        public Simulation(String unNom, int _vitesse)
        {
            Nom = unNom;
            vitesse = _vitesse;
            listePersonnages = new List<PersonnageAbstrait>();
            listeSujetsObserves = new List<SujetObserveAbstrait>();
            listeZones = new List<ZoneAbstraite>();
            listeConflits = new List<ConflitAbstrait>();
            listeObjets = new List<ObjetAbstrait>();
        }
        public void TourDeJeu()
        {
            foreach (SujetObserveAbstrait sujet in listeSujetsObserves)
            {
                sujet.Notifier();
            }
            foreach (PersonnageAbstrait perso in listePersonnages)
            {
                perso.AnalyserSituation();
                perso.Execution();
            }
            foreach (ConflitAbstrait conflit in listeConflits)
            {
                conflit.Mediation();
            }
            foreach (ObjetAbstrait objet in listeObjets)
            {
                objet.MiseAJour();
            }
            RecupererInformation();
            CalculStatistiques();
```

```
            //Afficher();
        }
        public void Afficher()
        {
            ModuleIHM.afficher();
        }
        public void Sauvegarder()
        {
        }
        public void RecupererInformation()
        {
            ModuleStats.RecupererInformation();
        }
        public void CalculStatistiques()
        {
            ModuleStats.CalculStatistiques();
        }


      public void chargerSimulation(String _emplacementFichierXML ){
                  ChargerSimulation chargement = new ChargerSimulation (this,
_emplacementFichierXML);
                  chargement.extractionDesDonnes ();
            }
    }
}
```

Thread principal

```
using ProjetDesignPattern.JeuDefenceTower;
using ProjetDesignPattern.JeuEchecs;
using ProjetDesignPattern.JeuSimulationTrafic;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
namespace ProjetDesignPattern
{
    public partial class FenPrincipale : Form
    {
        BackgroundWorker m_oWorker;
        Simulation jeu;
        public FenPrincipale()
        {
            InitializeComponent();
            m_oWorker = new BackgroundWorker();
            m_oWorker.DoWork += new DoWorkEventHandler(m_oWorker_DoWork);
            m_oWorker.ProgressChanged += new ProgressChangedEventHandler
                    (m_oWorker_ProgressChanged);
            m_oWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler
                    (m_oWorker_RunWorkerCompleted);
            m_oWorker.WorkerReportsProgress = true;
            m_oWorker.WorkerSupportsCancellation = true;
        }
        private void m_oWorker_RunWorkerCompleted(object sender,
RunWorkerCompletedEventArgs e)
        {
```

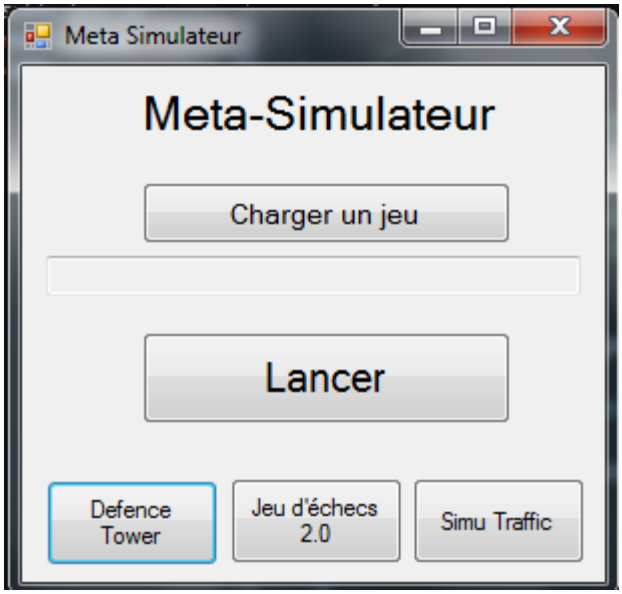
```
            MessageBox.Show("C'est la fin du jeu");
            if (jeu.Nom == "JeuDefenceTower")
            {
                ((ModuleIHMDT)jeu.ModuleIHM).ihm.Close();
            }
            if (jeu.Nom == "JeuSimulationTrafic")
            {
                ((ModuleIHM_Trafic)jeu.ModuleIHM).ihm.Close();
            }
            if (jeu.Nom == "JeuEchecs")
            {
                ((ModuleIHM_Echecs)jeu.ModuleIHM).ihm.Close();
            }
        }
        private void m_oWorker_ProgressChanged(object sender,
ProgressChangedEventArgs e)
        {
            jeu.Afficher();
        }
        private void m_oWorker_DoWork(object sender, DoWorkEventArgs e)
        {
            int i = 0;
            while(true && !jeu.finDuJeu)
            {
                System.Threading.Thread.Sleep(jeu.vitesse);
                jeu.TourDeJeu();
                m_oWorker.ReportProgress(i);
                i++;
            }

        }
    }
```

Fenêtre principale

Lancement

Au lancement du programme, une fenêtre se lance permettant de charger des fichiers XML pour lancer la simulation.



Après avoir chargé un fichier xml conforme, grâce au bouton “Charger le jeu”, le bouton “Lancer” permet de lancer la simulation chargée.
Les 3 boutons en bas du formulaire permettent de lancer des exemples de simulation déjà créées.