Projet C++ : "Simulation de Ville Virtuelle – VirtualCity -"

Bienvenue, architectes numériques ! : Vous avez été sélectionnés pour un projet passionnant qui permet de créer et gérer une ville virtuelle où tout est possible (ou presque).

Objectifs du projet

Créer une application en C++ avec une interface graphique (par exemple, en utilisant **Qt**) qui simule la gestion d'une ville virtuelle. Vous devez :

1. **Concevoir une ville virtuelle** avec différents bâtiments et habitants.
2. **Gérer les ressources de manière optimale** : eau, électricité, budget, pollution.
3. **Assurer le bien-être des habitants** en maintenant leur satisfaction à un niveau élevé.
4. **Survivre à des événements imprévus** qui peuvent bouleverser la vie de la ville.

Consignes Générales

1. **Travail en binôme**
2. **Structure du projet :**
    - Implémentez des classes bien organisées (avec headers séparés).
    - Respectez les principes de la programmation orientée objet (POO).
    - Utilisez l'héritage et la composition pour les relations entre classes.
3. **Remise du projet :**
    - Code source bien structuré.
    - Une présentation PowerPoint incluant des captures d'écran de votre simulation.
    - Un rapport technique expliquant votre architecture et vos choix.
    - Une démo de votre application pendant la soutenance.
4. **Deadline :** [Date envoyée avec le lancement du projet].

Détails Techniques

Les classes principales : Vous devez implémenter les classes suivantes :

1. Classe Batiment (classe de base): Représente un bâtiment générique.

- **Attributs principaux :**
  - id : Identifiant unique.
  - nom : Nom du bâtiment.
  - type : Type de bâtiment (Habitation, Ressource, Service, etc.).
  - consommationEau et consommationElectricite : Consommation par cycle.
  - effetSatisfaction : Impact sur la satisfaction des habitants.
- **Méthodes principales :**
  - afficherDetails() : Affiche les informations du bâtiment.
  - calculerImpactRessources() : Calcule l'impact sur les ressources de la ville.

2. Classe dérivée Maison: Représente une maison pouvant accueillir des habitants.

- **Attributs spécifiques :**
  - capaciteHabitants et habitantsActuels.
- **Méthodes spécifiques :**
  - ajouterHabitants(nb) et retirerHabitants(nb).

3. Classe dérivée Usine: Représente une usine produisant des ressources mais générant de la pollution.

- **Attributs spécifiques :**
  - productionRessources et pollution.
- **Méthodes spécifiques :**
  - produireRessources() et calculerPollution().

4. Classe dérivée Parc: Représente un parc qui améliore le bien-être des habitants.

- **Attributs spécifiques :**
  - surface et effetBienEtre.
- **Méthodes spécifiques :**
  - ameliorerBienEtre().

5. Classe Ville: Représente la ville entière avec ses habitants, bâtiments, et ressources.

- **Attributs principaux :**
  - nom, budget, population, satisfaction, batiments (liste de bâtiments).

- ressources : eau, électricité.

- **Méthodes principales :**
  - ajouterBatiment(Bâtiment b) et supprimerBatiment(id).
  - calculerConsommationTotale() et calculerSatisfaction().

6. Classe Simulation: Représente la logique de la simulation.

- **Attributs principaux :**
  - cycleActuel, ville, evenements.
- **Méthodes principales :**
  - demarrerCycle() et terminerCycle().
  - declencherEvenement().

Exigences Fonctionnelles

1. **Gestion des bâtiments :**
   - Construire différents types de bâtiments.
   - Supprimer ou modifier des bâtiments existants.
2. **Gestion des ressources :**
   - Assurer un équilibre entre production et consommation.
   - Prendre des décisions pour optimiser l'utilisation des ressources.
3. **Bien-être des habitants :**
   - Surveiller et maintenir un niveau élevé de satisfaction.
   - Ajouter des parcs ou des services pour améliorer le bien-être.
4. **Simulation d'événements :**
   - Implémentez des événements aléatoires (incendies, grèves, inondations).
   - Gérer leurs impacts sur les ressources et la satisfaction.

     Événements amusants :

     - Une nuée de pigeons géants envahit votre ville ! Résultat : une baisse de 15% de la satisfaction des habitants. De plus, ils déplorent la présence de trop de déchets dans les rues. Heureusement, vous avez un « service de nettoyage » à gérer pour remettre de l'ordre.
     - Une panne de courant fait grimper la consommation d'énergie de 50 %.
     - Les jardiniers de la ville sont en grève et refusent d'entretenir les parcs et espaces verts ! La satisfaction de vos habitants tombe de 20% et l'effet des parcs diminue de moitié.

- Une panne géante de transports publics se produit, tous les bus de la ville sont bloqués par un gigantesque embouteillage ! Résultat : la satisfaction chute brutalement. Les habitants doivent maintenant marcher pour aller au travail, à l'école et au parc.

- Une tempête de neige imprévu bloque les routes, interrompt la production des usines et rend les habitants grumpy. Mais ne vous inquiétez pas, vous avez une chance d'envoyer des équipes de déneigement pour sauver la situation.

Barème de Notation

- **Conception des classes :** 10%
- **Fonctionnalités implémentées :** 50%
- **Qualité du code (propreté, commentaires) :** 20%
- **Qualité du rapport et de la présentation :** 10%
- **Originalité et créativité :** 10%

Avec un peu de code et beaucoup d'imagination,
votre ville pourrait devenir une métropole prospère...
ou un véritable chaos ! 😅

Bonne chance !