# Créer un jeu pour ARCADE

Guide complet pour développer un jeu compatible avec l'architecture ARCADE

## Introduction

L'architecture ARCADE permet de développer des jeux modulaires qui fonctionnent avec différentes bibliothèques graphiques. Voici comment créer votre propre jeu pour la plateforme ARCADE en respectant l'interface définie.

## Interface IGameModule

### Comprendre l'interface

Tous les jeux ARCADE doivent implémenter l'interface `IGameModule` qui définit les méthodes essentielles pour l'intégration avec le core.

```
class IGameModule {
public:
    virtual ~IGameModule() = default;
    virtual void init() = 0;
    virtual void processEvent(GameEvent event) = 0;
    virtual void update() = 0;
    virtual bool isGameOver() const = 0;
    virtual std::string getName() const = 0;
    virtual int getScore() const = 0;
    virtual bool isPaused() const = 0;
    virtual void reset() = 0;
    virtual std::vector<int> getGrid() const = 0;
    virtual std::vector<int> getEntities() const = 0;
};
```

Copy

### Méthodes principales

- **init()** - Initialise le jeu et ses composants
- **processEvent(GameEvent)** - Traite les événements du joueur
- **update()** - Met à jour la logique du jeu
- **isGameOver()** - Indique si la partie est terminée
- **getGrid()** - Retourne la grille de jeu pour l'affichage
- **getEntities()** - Retourne les entités du jeu pour un rendu plus avancé

## Structure d'un jeu

### Fichiers nécessaires

Pour créer un jeu compatible, vous aurez besoin des fichiers suivants:

- **MonJeu.hpp** - Définition de la classe de jeu
- **MonJeu.cpp** - Implémentation de la logique du jeu
- **MonJeuModule.cpp** - Point d'entrée pour le chargement dynamique

### Point d'entrée

Le fichier module doit implémenter les fonctions d'exportation pour le chargement dynamique:

```
extern "C" {
    arcade::IGameModule *createGame()
    {
        return new arcade::MonJeu();
    }

    void destroyGame(arcade::IGameModule *game)
    {
        if (game)
            delete game;
    }
}
```

Copy

## Exemple complet: Snake

### Définition de la classe

Exemple de définition de classe pour un jeu Snake:

```
class Snake : public IGameModule {
public:
    Snake();
    ~Snake() = default;

    void init() override;
    void update() override;
    void processEvent(GameEvent event) override;

    bool isGameOver() const override { return _gameOver; }
    int getScore() const override { return _score; }
    std::string getName() const override { return "Snake"; }
    bool isPaused() const override { return _isPaused; }
    std::vector<int> getGrid() const override;
    std::vector<int> getEntities() const override { /* ... */ }

private:
    static const int GRID_WIDTH = 20;
    static const int GRID_HEIGHT = 15;

    std::deque _snake;
    Position _food;
    Direction _direction;
    Direction _pendingDirection;

    bool _gameOver;
    bool _isPaused;
    int _score;
    double _speed;

    void spawnFood();
    void moveSnake();
    void reset();
    // Autres méthodes internes...
};
```

Copy

Email: nopperoductions@gmail.com
WhatsApp: Enviez-nous ici