Documentation du projet Arcade

by : Léo Paol, Charles Paulet, Nicolas-Emmanuel Robert, Théophile Champion, Clovis Peridy, Hugo Ailleres

Somaire

Les Manageurs

LibManager

Le LibManager vas s'occuper de charger les différentes librairies, et appeler les bonnes fonctions.

```
class LibManager
{
public:
  LibManager(core *_core);

private:
    std::map<std::string, IGraph *>   _maps;
    // on récupère les pointeurs de chaque librairie
};
```

GameManager

Le GameManager vas s'occuper de charger les jeux, puis d'intéragir avec ceux-ci.

```
class GameManager
{
public:
  GameManager(core *_core);

private:
  std::map<std::string, IGame *>    _games;
};
```

EventManager

```
class EventManager
{
public:
```

```
  EventManager(core *_core);

private:
  std::queue<eventType>   _event;
  // queue d'events founi par IGraph puis interpreter par le core
  // ensuite celui-ci enpile et dépile en fontion du core
};
```

Le core charge les trois managers et s'occupe de syncroniser ceux-ci.

Les class

eventTypes.hh

```
enum eventTypes
{
  KEY_UP,
  KEY_DOWN,
  KEY_LEFT,
  KEY_RIGHT,
  KEY_PREV_GRAPH,
  KEY_NEXT_GRAPH,
  KEY_PREV_GAME,
  KEY_NEXT_GAME,
  KEY_RESTART_GAME,
  KEY_GO_BACK_MENU
};
```

tile.hh

```
class Tile
{
public:
  Tile(int x, int y, std::string texture_name); // x, y position sur grille

private:
  int           x;
  int           y;
  std::string   texture_name;
  // le game créer un tableau de tuiles puis les envoie à la lib graphique qui vas l'interpr
};
```

IGraph.hh

```
class IGraph
{
  virtual ~IGraph() {}
  virtual void  up() = 0;
  virtual void  down() = 0;
  virtual void  print(const vector<const Tile> *) const = 0;
  virtual void  pull() = 0;
};
```

IGame.hh

```
class IGame
{
public:
  virtual                           ~IGame() {}
  virtual void                      restart() = 0;
```

```
  virtual const vector<const Tile>*   simulate() = 0;
  virtual void                        handleEvent(const EventType) = 0;
  virtual const std::string           &getName() const = 0;
};
```