



Arcade

Création de modules graphiques et de jeux

Création d'un module graphique

Organisation des fichiers

Les modules graphiques se trouvent dans le dossier *graphical_libs*.

Pour créer votre module graphique, créez un sous dossier portant le nom de votre module.

A l'intérieur, créez-y un Makefile reprenant ce modèle (en remplaçant la target par le nom de votre module) :

```
CC      =      g++
TARGET  =      ../../lib/arcade_name_of_the_module.so
CPPFLAGS =      -W -Wall -Wextra -std=c++17 -fPIC
LDFLAGS =      -shared
SRC      =      $(wildcard *.cpp)      \
                ../../core/AGraphicalModule.cpp
OBJ      =      $(SRC:.cpp=.o)
all:     $(OBJ)
          $(CC) $(OBJ) -o $(TARGET) $(LDFLAGS)
clean:
          $(RM) $(OBJ)
fclean:  clean
          $(RM) $(TARGET)
re:      fclean all
```

Ensuite, créez une classe héritant de *AGraphicalModule* contenue dans *core/AGraphicalModule.hpp*.

Votre classe devra être contenue dans le namespace *Arcade*.

Structure de données Organisation des

Création d'un module graphique

Méthodes de la classe

Votre classe devra obligatoirement surcharger les méthodes suivantes :

Affichage d'un sprite à l'écran

```
void drawSprite(graphical_sprite_t &sprite);
```

Affichage d'un texte à l'écran

```
void drawText(graphical_text_t &text);
```

Affichage d'un carré à l'écran

```
void showInputBox(graphical_box_t &box);
```

Retourne true ou false selon si la souris est cliquée ou non

```
bool isMouseClicked();
```

Retourne la position de la souris

```
graphical_vector_t getMousePosition();
```

Détruit tous les éléments en mémoire

```
void reset();
```

Vide le buffer d'affichage

```
void clear();
```

Actualise l'affichage

```
void refresh();
```

Met à jour le tableau de touches `_keys`

```
void updateInputsMap();
```

Ouvre la fenêtre

```
void openWindow();
```

Ferme la fenêtre

```
void closeWindow();
```

Création d'un module graphique

Structure de données

Les méthodes définies ci-dessus utilise des structures de données spécifiques au programme.

graphical_sprite_t

unsigned int **id** -> Identifiant unique du sprite (permet de mettre en place un système de cache)

std::string **path** -> Chemin de la texture (image au format PNG)

bool **ncursesBox** -> Ignorer

graphical_color_t **color** -> Couleur du sprite

graphical_vector_t **pos** -> Position du sprite

graphical_vector_t **size** -> Taille du sprite

float **angle** -> Rotation du sprite

bool **visible** -> Visibilité du sprite

graphical_text_t

unsigned int **id** -> Identifiant unique du texte (permet de mettre en place un système de cache)

std::string **text** -> Texte à afficher

graphical_vector_t **pos** -> Position du texte

int **size** -> Taille de la police

graphical_color_t **color** -> Couleur du texte

std::string **font** -> Chemin de la police

graphical_box_t

graphical_vector_t **pos** -> Position du carré

graphical_vector_t **size** -> Taille du carré

std::string **input** -> Texte à afficher dans le carré

Organisation des fichiersStructure de

Création d'un jeu

Organisation des fichiers

Les jeux se trouvent dans le dossier *game_libs*.

Pour créer votre module graphique, créez un sous dossier portant le nom de votre module.

A l'intérieur, créez-y un Makefile reprenant ce modèle (en remplaçant la target par le nom de votre module) :

```
CC      =      g++
TARGET  =      ../../lib/arcade_name_of_the_game.so
CPPFLAGS =      -W -Wall -Wextra -std=c++17 -fPIC
LDFLAGS =      -shared
SRC      =      $(wildcard *.cpp)      \
                ../../core/AGraphicalModule.cpp
                ../../core/GameClock.cpp
OBJ      =      $(SRC:.cpp=.o)
all:      $(OBJ)
           $(CC) $(OBJ) -o $(TARGET) $(LDFLAGS)
clean:
           $(RM) $(OBJ)
fclean: clean
           $(RM) $(TARGET)
re:      fclean all
```

Ensuite, créez une classe héritant de *AGameModule* contenue dans *core/AGameModule.hpp*.

Votre classe devra être contenue dans le namespace *Arcade*.

Création d'un jeu

Méthodes de la classe

Votre classe devra obligatoirement surcharger les méthodes suivantes :

Lancement du jeu

```
void startGame() ;;
```

Affichage des éléments à l'écran, utiliser la liste fournie en paramètre

```
int updateGame(std::list<std::pair<Arcade::FLAGS, IStruct_t *>> *_list);
```

La liste fournie en paramètre est celle qui doit contenir tous les éléments graphiques à afficher.

Exemple :

```
list->push_back(std::make_pair(BOX, &_box));  
for (auto &n : _texts)  
list->push_back(std::make_pair(TEXT, &n));  
for (auto &n : _powerUps)  
list->push_back(std::make_pair(SPRITE, &n.first));  
for (auto &n : _sprites)  
list->push_back(std::make_pair(SPRITE, &n));  
for (auto &n : _ennemiesShots)  
list->push_back(std::make_pair(SPRITE, &n.first));
```