|  |
| --- |
|  |
| Documentation pour l’Arcade |
|  |

|  |
| --- |
| Geoffrey Davril, Dylan Deu, Antoine Morel, Baptiste Mairesse  01/04/2017 |

Table des matières

[Arcade – quelques mots 3](#_Toc479503438)

[Bibliothèques de jeux 4](#_Toc479503439)

[Créer une bibliothèque 4](#_Toc479503440)

[Protocole de communication 4](#_Toc479503441)

[GetMap 4](#_Toc479503442)

[Position 5](#_Toc479503443)

[WhereAmI 5](#_Toc479503444)

[Interface IGame 5](#_Toc479503445)

[putAction 5](#_Toc479503446)

[getMap 5](#_Toc479503447)

[getWhereAmI 5](#_Toc479503448)

[getScore 5](#_Toc479503449)

[getName 6](#_Toc479503450)

[deactivateTimer 6](#_Toc479503451)

[Bibliothèques graphiques 6](#_Toc479503452)

[Créer une bibliothèque 6](#_Toc479503453)

[Interface IWindow 7](#_Toc479503454)

[Close 7](#_Toc479503455)

[drawPixel 7](#_Toc479503456)

[drawSquare 7](#_Toc479503457)

[drawText 8](#_Toc479503458)

[getLastEvent 8](#_Toc479503459)

[display 8](#_Toc479503460)

[Interface IPosition 8](#_Toc479503461)

[getX 8](#_Toc479503462)

[getY 8](#_Toc479503463)

[Interface IPixel 8](#_Toc479503464)

[getPosition 8](#_Toc479503465)

[getColor 8](#_Toc479503466)

[Interface IText 9](#_Toc479503467)

[getPosition 9](#_Toc479503468)

[getColorFore 9](#_Toc479503469)

[getColorBack 9](#_Toc479503470)

[getText 9](#_Toc479503471)

[Interface IEvent 9](#_Toc479503472)

[getType 9](#_Toc479503473)

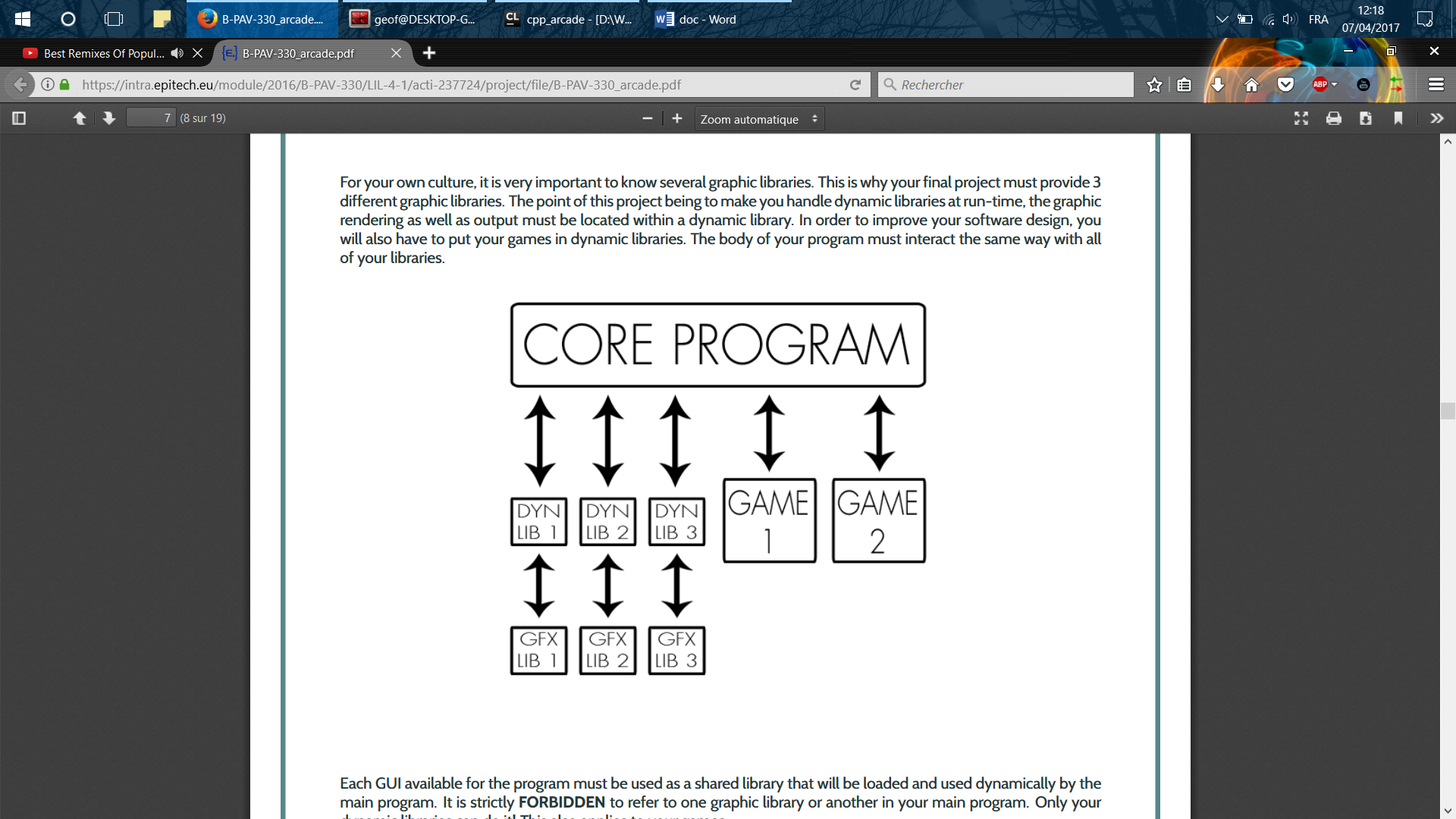
[Interface IMouseEvent 9](#_Toc479503474)

[getContent 9](#_Toc479503475)

[Interface IKeyboardEvent 10](#_Toc479503476)

[getContent 10](#_Toc479503477)

# Arcade – quelques mots



Le projet Arcade a pour but, la création de bibliothèques graphiques ainsi que de bibliothèques de jeux. Chacune de ces bibliothèques doivent être compatibles avec un autre groupe. Afin de rendre nos bibliothèques compatibles, nous avons dû nous mettre d’accord sur une architecture commune dans l’objectif d’avoir des classes implémentant ayant des comportements similaires.

# Bibliothèques de jeux

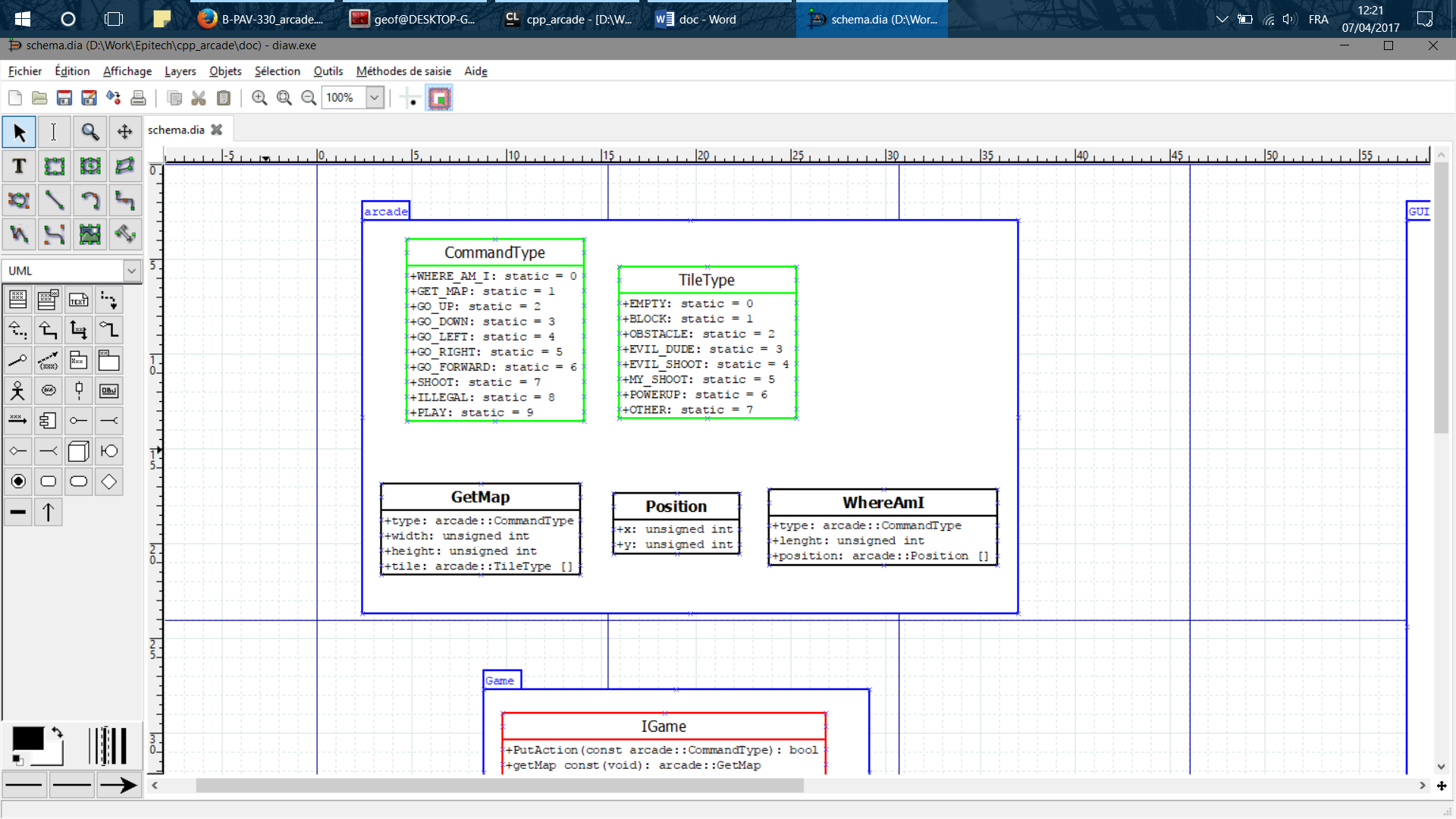
## Créer une bibliothèque

Pour créer une bibliothèque de jeu, la méthode *createGame* doit être implémentée. Cette méthode retourne un nouvel objet *Game* comportant les méthodes suivantes :

* putAction (const arcade::CommandType ) : bool
* getMap const (void) : arcade::GetMap
* getWhereAmI const (void) : arcade::WhereAmI
* getScore const (void) : unsigned long int
* getName const (void) : std::string
* deactivateTimer (void) : void

Toutes ces méthodes sont décrites ci-dessous.

## Protocole de communication



### GetMap

Cette classe permet de décrire la carte du jeu à partir de sa largeur (width) et de sa hauteur (height).

La carte est contenue dans l’attribut tile de la classe. Il s’agit d’un tableau de TileType. Ces derniers représentent le contenu d’un bloc de la carte.

La carte représente un repère orthogonal d’origine (y ; x) = (0 ; 0). Cette origine représente le coin haut gauche de la carte. L’axe Oy est orienté vers le vas et s’incrémente d’un pour accéder à la ligne suivante. L’axe Ox est orienté vers la droite et s’incrémente aussi d’un pour accéder à la colonne suivante.

On accède à un bloc de coordonnées (y, x) de la carte en accédant à l’élément *y \* largeur + x* de l’attribut tile.

### Position

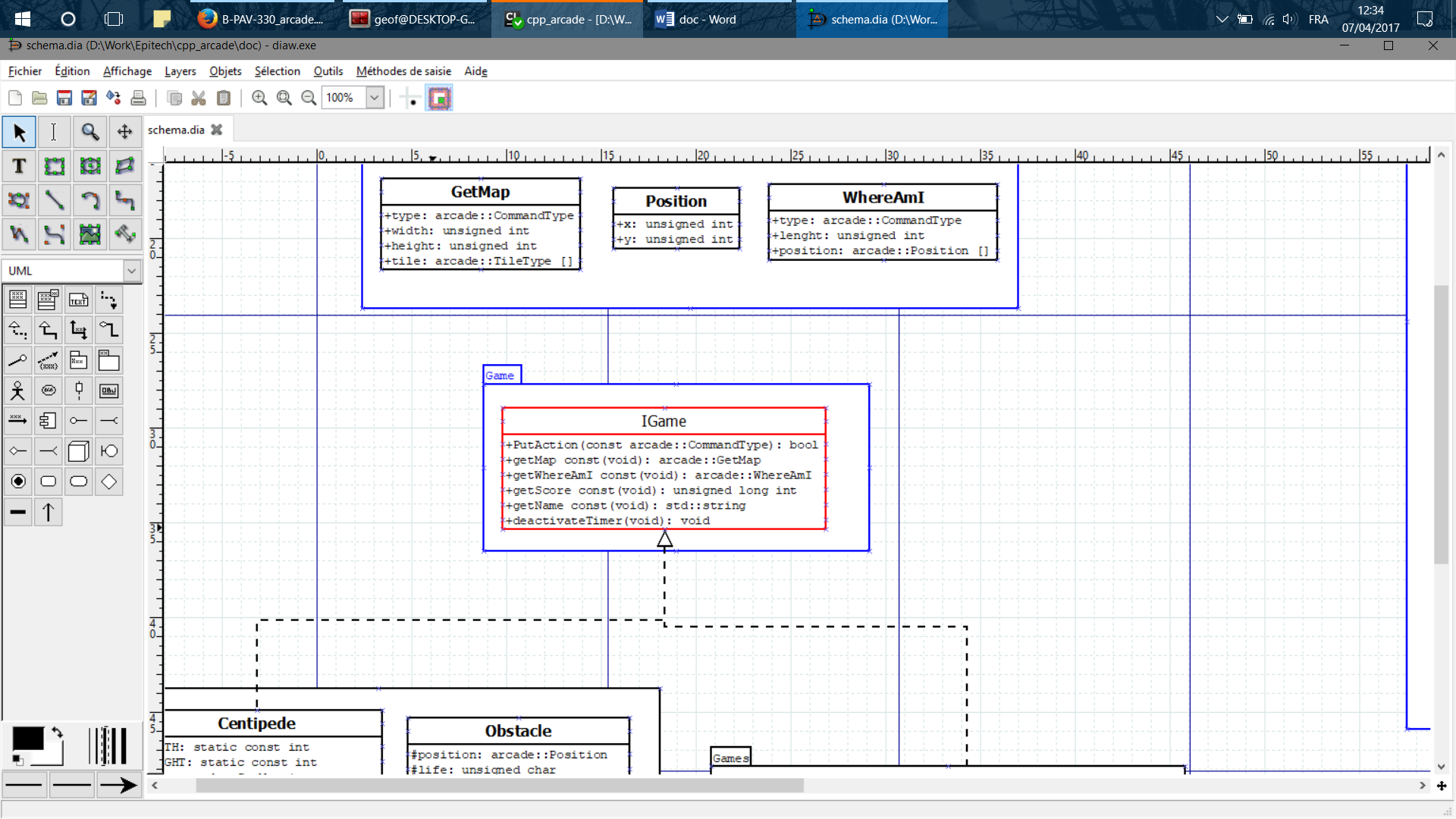
Cette classe indique une position (y ; x) dans la carte du jeu

### WhereAmI

Cette classe représente le joueur et sa position sur la carte.

Sa position est déterminée à partir de l’attribut position de la classe, qui est un tableau d’objets *Position* de longueur *length*.

## Interface IGame



L’interface IGame permet de décrire le comportement d’un jeu.

### putAction

Cette méthode prend en paramètre un *CommandType*. Elle doit être appelée dans le programme principal avant chaque rafraichissement de fenêtre. Elle permet d’envoyer à la bibliothèque de jeu une éventuelle action à effectuer ou une demande de rafraîchissement (*CommandType ::PLAY*).

Cette méthode retourne *false* si le jeu s’est terminé d’une façon ou d’une autre. Elle revoie *true* dans le cas contraire.

### getMap

Cette méthode retourne la carte de jeu.

### getWhereAmI

Cette méthode retourne la position du joueur.

### getScore

Cette méthode retourne le score du joueur.

### getName

Cette méthode retourne le nom du jeu.

### deactivateTimer

Cette méthode permet de désactiver la gestion du temps au sein du jeu.

Si elle est appelée, la bibliothèque de jeu doit alors considérer que c’est le programme principal qui gère le temps. Alors, la bibliothèque doit considérer chaque appel à *putAction* comme un tour suivant dans le jeu.

# Bibliothèques graphiques

## Créer une bibliothèque

Pour créer une bibliothèque graphique, les méthodes suivantes doivent êtres implémentées :

* GUI::IWindow &(\*createWindow)(const std::string, int, int);
* GUI::IPosition &(\*createPosition)(unsigned int, unsigned int);
* GUI::IPixel &(\*createPixel)(const GUI::IPosition &, const GUI::Color &);
* GUI::IText &(\*createText)(const std::string &, const GUI::Color &, const GUI::Color &, const GUI::IPosition &);

createWindow retourne un nouvel objet Window ayant les méthodes suivantes :

* virtual bool drawPixel(const GUI::IPixel &pixel);
* virtual bool drawSquare(const GUI::IPosition &posBegin,
* const GUI::IPosition &posEnd,
* const GUI::Color &color);
* virtual bool drawText(const GUI::IText &text);
* virtual const GUI::Events::IEvent \*getLastEvent();
* virtual bool display();

Ces méthodes sont décrites ci-dessous.

createPosition retourne un nouvel objet Position ayant les méthodes suivantes :

* virtual unsigned int getX() const;
* virtual unsigned int getY() const;
* void setY (unsigned int y);
* void setX (unsigned int x);

Ces méthodes sont décrites ci-dessous.

createPixel retourne un nouvel objet ayant les méthodes suivantes :

* virtual const GUI::IPosition &getPosition() const;
* virtual const GUI::Color &getColor() const;
* void setColor(const GUI::Color &);
* void setPos(const unsigned int, const unsigned int);

Ces méthodes sont décrites ci-dessous;

createText retourne un nouvel objet Text ayant les méthodes suivantes :

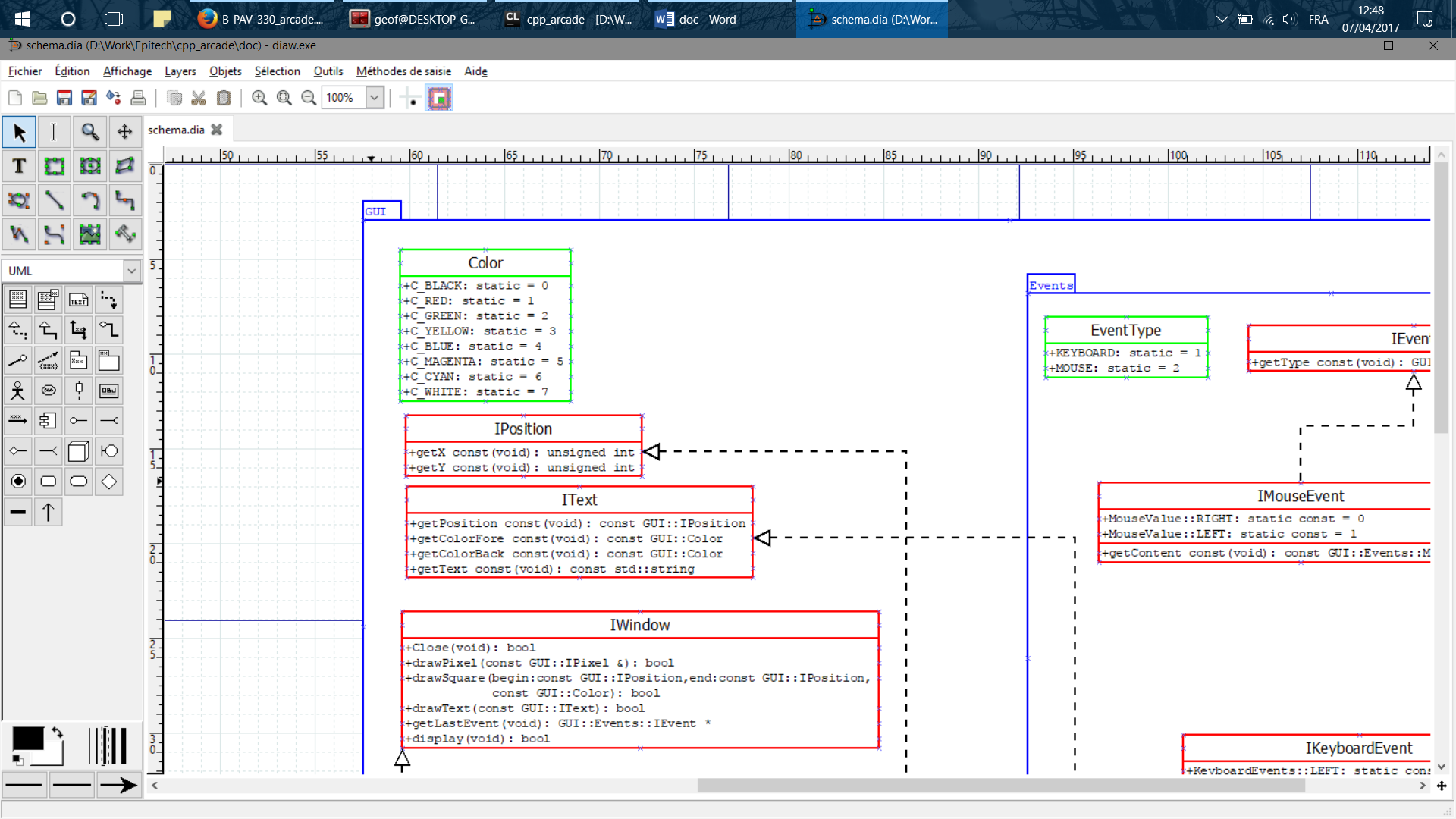
* virtual const GUI::IPosition &getPosition() const;
* virtual const GUI::Color &getColorFore() const;
* virtual const GUI::Color &getColorBack() const;
* virtual const std::string &getText() const;
* void setText(const std::string &);

Ces méthodes sont décrites ci-dessous.

Dans les bibliothèques graphiques, on considère pour taille d’une unité de mesure, la taille de représentation d’un bloc de jeu. Ainsi, un pixel ne sera pas représenté par un unique pixel sur l’écran, mais par un carré de pixels.

Comme pour une carte de jeu (classe GetMap), on considère que le coin haut gauche est l’origine de la fenêtre et que l’axe Ox a pour direction la droite et Oy le bas.

## Interface IWindow



L’interface IWindow permet de décrire le comportement d’une fenêtre graphique.

### Close

Cette méthode permet d’indiquer que la fenêtre doit être fermée.

Retourne *true* si aucun problème n’a été relevé.

### drawPixel

Cette méthode permet de dessiner un pixel dans la fenêtre.

Elle prend en paramètre une instance de IPixel, qui contient les caractéristiques du pixel à dessiner.

Retourne *true* si aucun problème n’a été relevé.

### drawSquare

Cette méthode permet de dessiner un rectangle dans la fenêtre.

Elle prend en paramètre deux instances de IPosition : une première qui correspond au coin haut gauche du carré et une seconde à son coin bas droit. L’ensemble des pixels du rectangle induit par ces coordonnées seront dessiné dans la couleur précisée en dernier paramètre.

Retourne *true* si aucun problème n’a été relevé.

### drawText

Cette méthode permet d’écrire du texte dans la fenêtre.

Elle prend en paramètre une instance de IText, qui contient les caractéristiques du texte à écrire.

Retourne *true* si aucun problème n’a été relevé.

### getLastEvent

Retourne une instance IEvent correspondant au dernier événement graphique qui s’est produit.

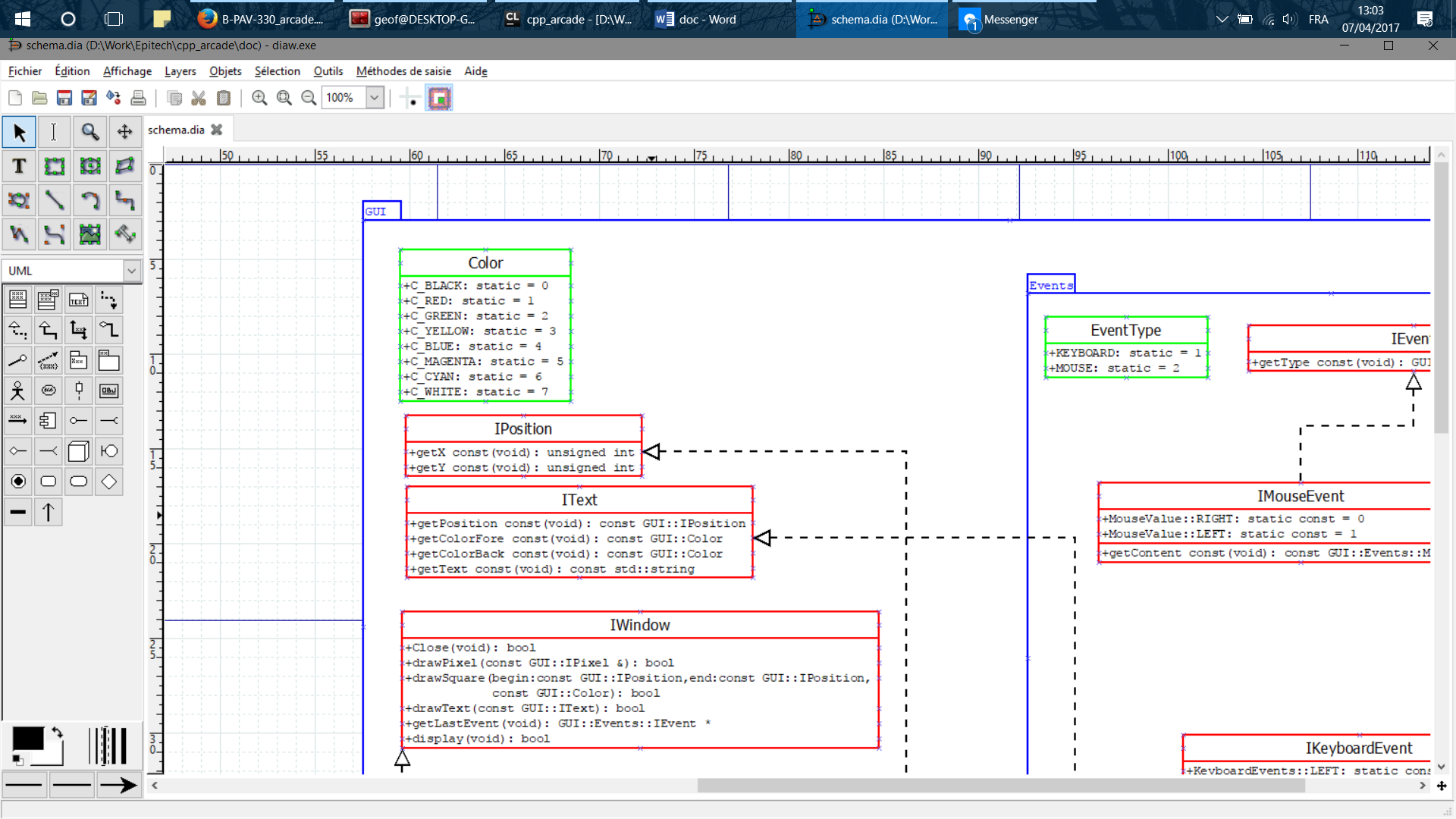
Retourne un pointeur nul si aucun événement ne s’est produit.

### display

Ordonne le nettoyage de la fenêtre et l’application dans cette dernière des éléments demandés par les méthodes drawPixel, drawSquare et drawText.

Retourne *true* si aucun problème n’a été relevé.

## Interface IPosition



Cette interface représente une position dans la fenêtre graphique.

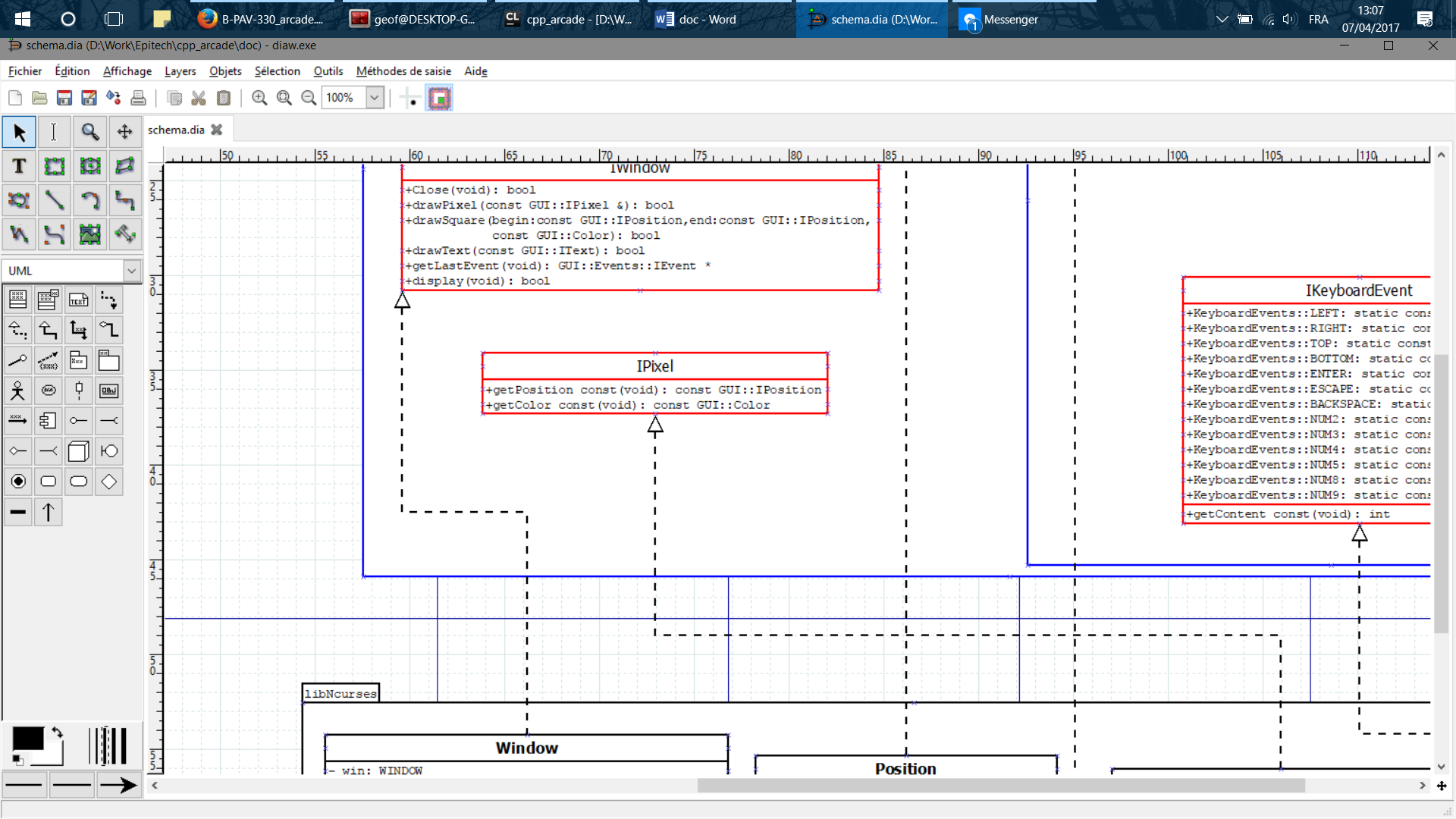
### getX

Retourne la position dans la fenêtre sur l’axe horizontal.

### getY

Retourne la position dans la fenêtre sur l’axe vertical.

## Interface IPixel



Cette interface décrit le comportement des objets permettant de représenter les pixels à dessiner dans la fenêtre graphique.

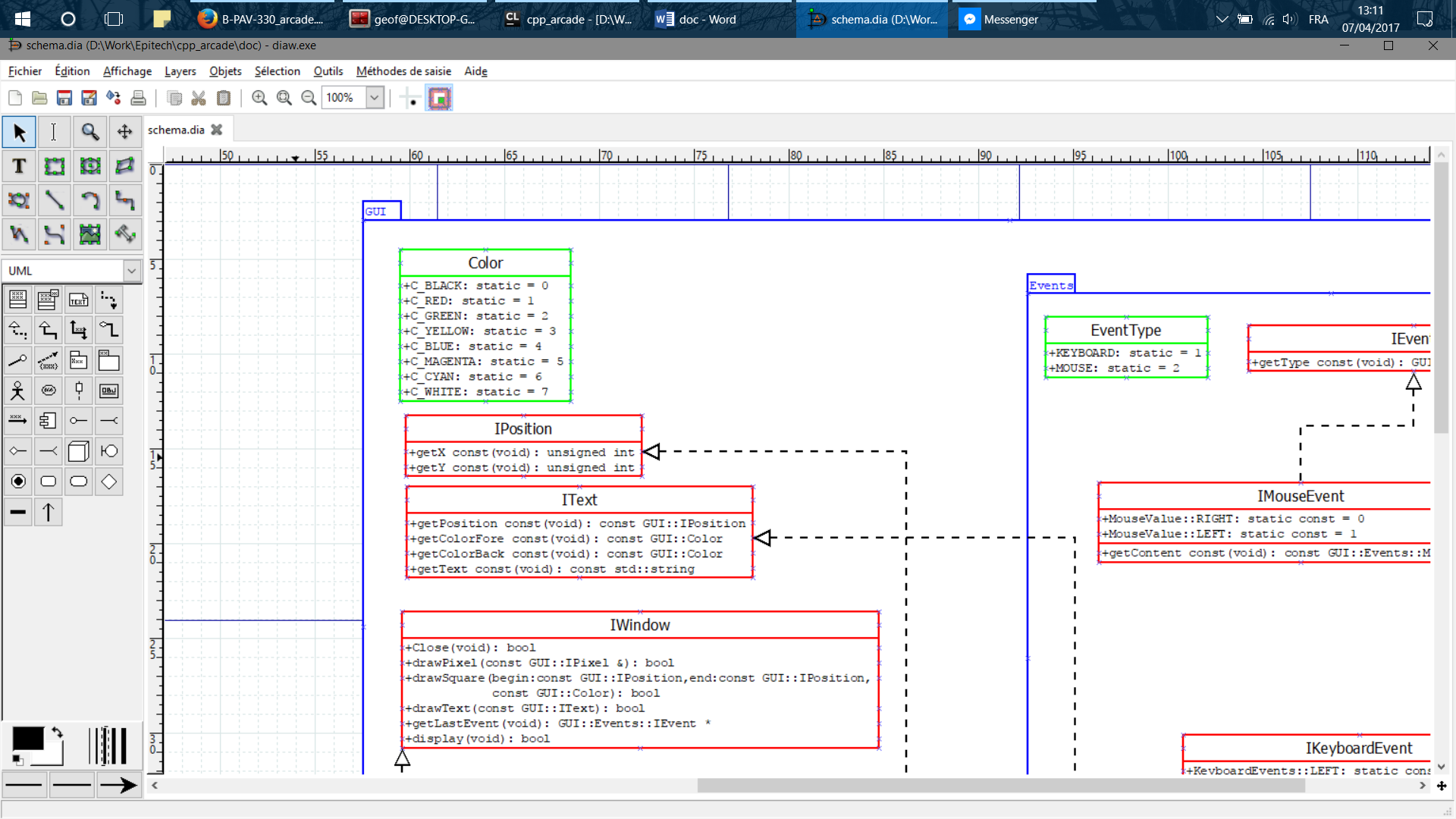
### getPosition

Retourne une instance de IPosition décrivant l’emplacement dans la fenêtre du pixel.

### getColor

Retourne la couleur du pixel.

## Interface IText



Cette interface permet de décrire le comportement d’un objet texte.

### getPosition

Retourne la position dans la fenêtre où le texte commence à être écrit.

### getColorFore

Retourne la couleur du texte

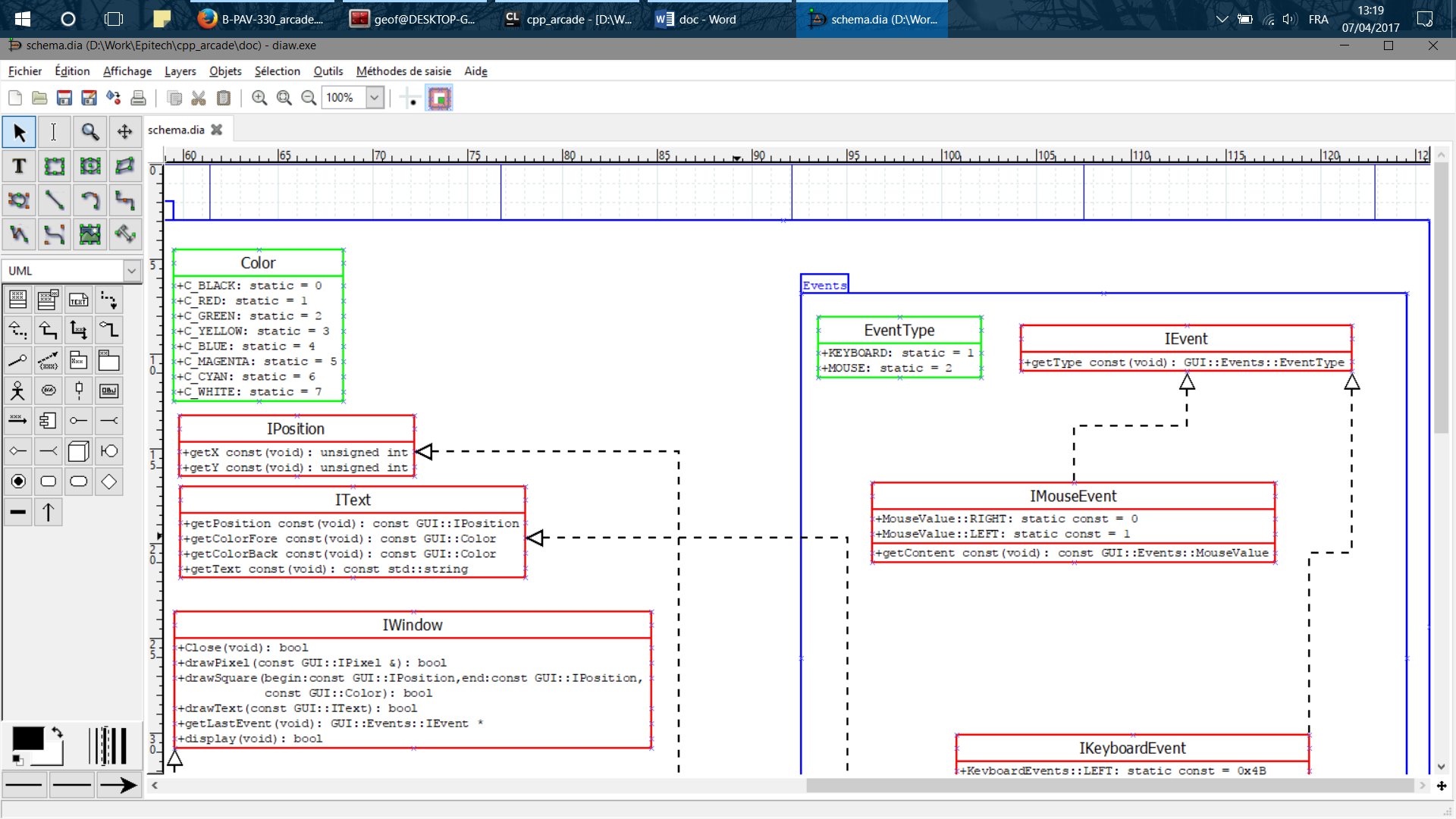
### getColorBack

Retourne la couleur du fond du texte.

### getText

Retourne le contenu du texte.

## Interface IEvent

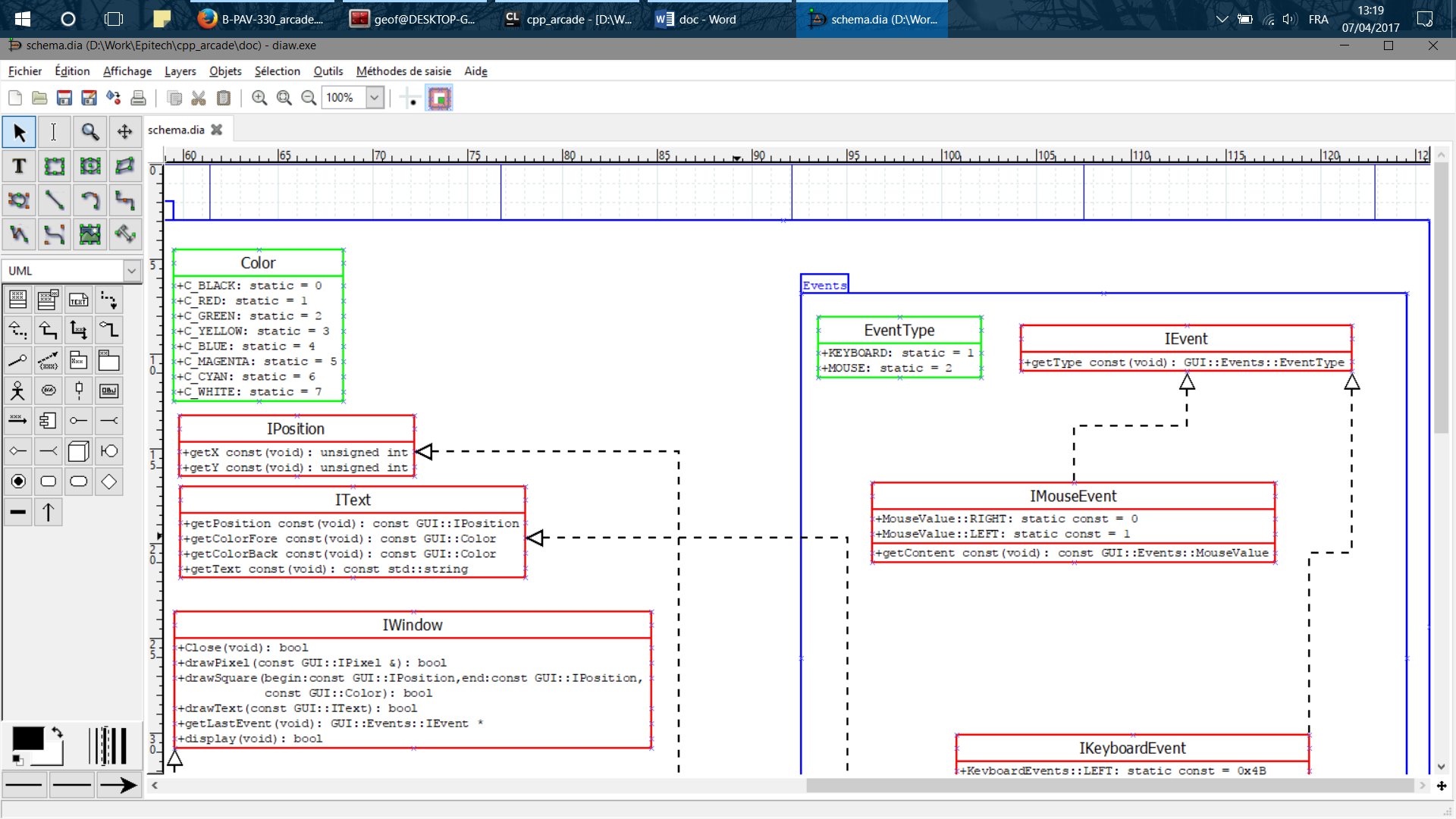


Cette interface est le type de tout événement graphique. Permet d’un déterminer son type.

### getType

Retourne le type de l’événement (KEYBOARD ou MOUSE).

## Interface IMouseEvent



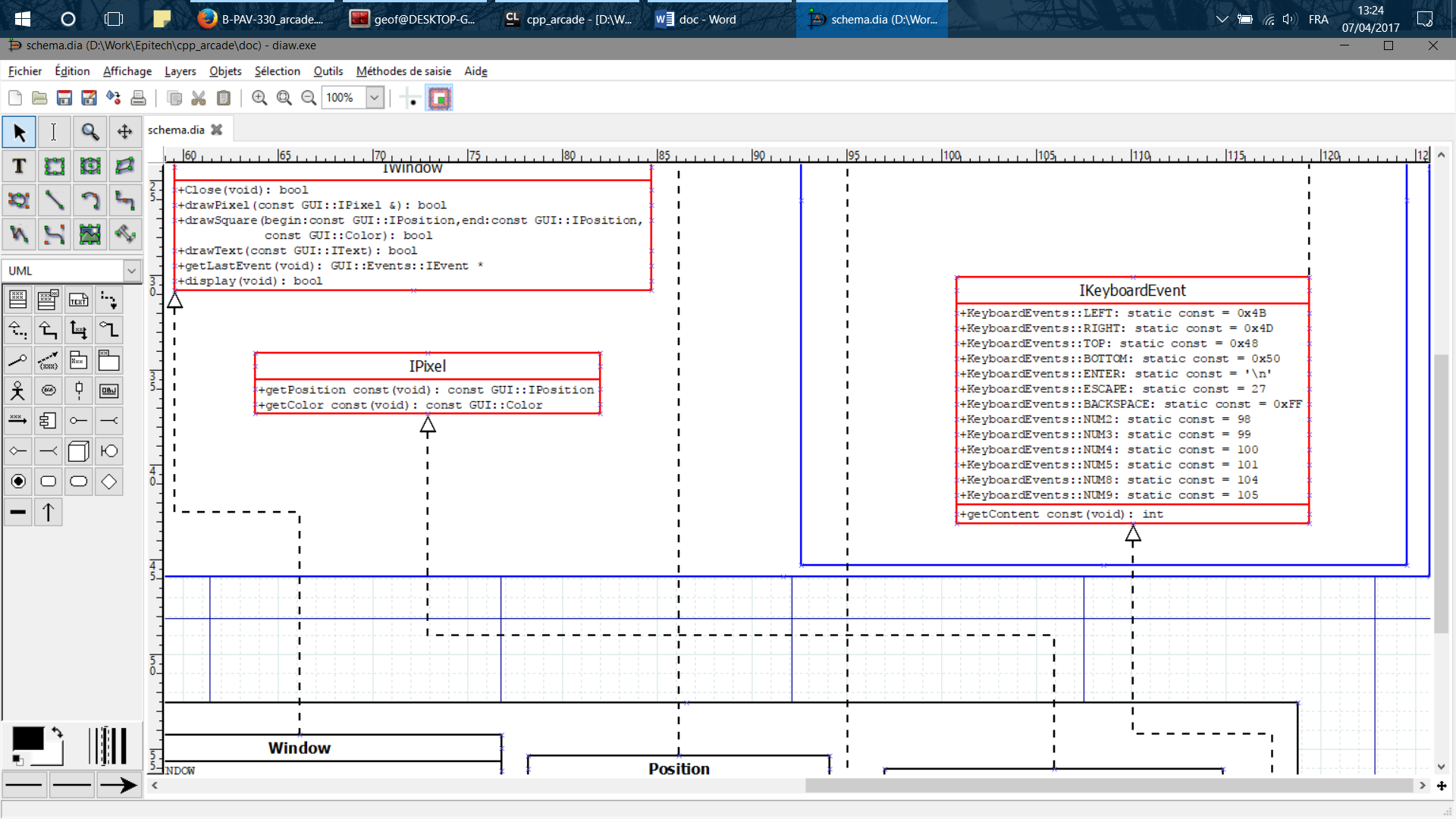
Cette interface décrit un événement souris.

Hérite de IEvent

### getContent

Retourne le contenu de l’événement souris associé.

## Interface IKeyboardEvent



Cette interface décrit un événement clavier.

Hérite de IEvent

### getContent

Retourne le contenu de l’événement clavier associé.