ANNEXE UTILISATION DES NOTIONS DU COURS

I. Chapitre 1

I.1. Surcharge de fonction

Fichier / Numéros de ligne : `include/tangram/geometry/Point.hpp` (l. 48, 52, 59, 61)

```
template<typename U>
Point<T> operator*(const Point<U> &other) const;

template<typename U>
Point<T> operator*(U factor) const;
```

II. Chapitre 2

II.1. Attribut static

Fichier / Numéros de ligne : `include/tangram/game/Engine.hpp` (l. 20-27)

```
static const std::string DEFAULT_SHAPE_PATH = "../resources/default.shp";

static const std::string BACKGROUND_DIR = "../resources/background/";

static const std::string SHAPE_DIR = "../resources/shapes/";

static const std::string FONT_DIR = "../resources/fonts/";
```

II.2. Méthode static

Fichier / Numéros de ligne : `include/tangram/geometry/Shape.hpp` (l. 36)

```
static Shape load(const std::string &path);
```

III. Chapitre 3 : Surcharge d'opérateur

III.1. Surcharge d'opérateur (hors « et »)

Fichier / Numéros de ligne : `include/tangram/geometry/Point.hpp` (l. 52-67)

```
T &operator[](int i);

bool operator==(const Point<T> &other) const;

Point<T> operator+(const Point<T> &other) const;

Point<T> operator-(const Point<T> &other) const;

int32_t operator^(const Point<T> &other) const;

template<typename U>
Point<T> operator*(const Point<U> &other) const;
```

IV. Chapitre 4 : Héritage, Polymorphisme

IV.1 Fonctions virtuelles

Fichier / Numéros de ligne : `include/tangram/gui/Drawable.hpp` (interface)

```
virtual ~Drawable() = default;

virtual void draw() const = 0;
```

IV.2 Override

Fichier / Numéros de ligne : `include/tangram/state/Menu.hpp` (l. 35-43)

```
void cleanup() override;

void pause() override;

void resume() override;

bool update(const game::Event &event, game::Engine &engine) override;

void draw() const override;
```

IV.3 Final (classe, méthode)

Fichier / Numéros de ligne : `include/tangram/gui/ButtonDraw.hpp` (l.41)

```
void draw() const final;
```

V. Chapitre 5 : Template, STL

V.1. Classe template

Fichier / Numéros de ligne : `include/tangram/geometry/Point.hpp` (Tout le fichier ?)

```
template<typename T>
class Point {
        static_assert(std::is_arithmetic<T>::value, "Arithmetic type is required");

    public:
        static constexpr uint16_t NEAR_THRESHOLD = 15;

        T x = static_cast<T>(0);
        T y = static_cast<T>(0);


        ////////////////////// CONSTRUCTOR & DESTRUCTOR //////////////////////

        Point() = default;

        Point(T x, T y);

        template<typename U>
        Point<T>(const Point<U> &p);


        ////////////////////////////// OPERATORS //////////////////////////////

        T &operator[](int i);

        bool operator==(const Point<T> &other) const;
```

```
        Point<T> operator+(const Point<T> &other) const;

        Point<T> operator-(const Point<T> &other) const;

        int32_t operator^(const Point<T> &other) const;

        template<typename U>
        Point<T> operator*(const Point<U> &other) const;

        template<typename U>

        Point<T> operator*(U factor) const;

        template<typename U>
        friend std::ostream &operator<<(std::ostream &os, const Point<U> &p);


        ////////////////////////////// OTHERS ////////////////////////////////

        [[nodiscard]] Point<T> rotate(int16_t angle, const Point<double> &center) const;

        [[nodiscard]] Point<T> translate(const Point<T> &translation) const;

        [[nodiscard]] static Point<double> center(const std::vector<Point<T>> &points);

        [[nodiscard]] static Point<double> center(Point<T> first);

        template<typename... Args>
        [[nodiscard]] static Point<double> center(Point<T> first, Args... args);
};
```

V.2 Classes STL

VI.2.a. Vector Fichier / Numéros de ligne : `include/tangram/state/Menu.hpp` (l. 25,26)

```
std::vector<std::shared_ptr<game::Updatable>> updatables = {};
std::vector<std::shared_ptr<gui::Drawable>> drawables = {};
```

VI.2.b. Unordered_map Fichier / Numéros de ligne : `include/tangram/game/Updatable.hpp` (l. 27,28)

```
std::unordered_map<MLV_Mouse_button, InputState> buttons;
std::unordered_map<MLV_Keyboard_button, InputState> keys;
```

VI.2.c. String Fichier / Numéros de ligne : `include/tangram/game/Engine.hpp` (l. 20-23)

```
static const std::string DEFAULT_SHAPE_PATH = "../resources/default.shp"; /**< Path to the default shap
static const std::string BACKGROUND_DIR = "../resources/background/"; /**< Path to the directory contai
static const std::string SHAPE_DIR = "../resources/shapes/"; /**< Path to the directory containing save
static const std::string FONT_DIR = "../resources/fonts/"; /**< Path to the directory containing fonts
```

VI.2.d. Stack Fichier / Numéros de ligne : `include/tangram/game/Engine.hpp` (l. 35)

```
std::stack<state::State *> states = {};
```

VI. Chapitre 6 : Foncteur

VI.1. Foncteur

Fichier / Numéros de ligne : `include/tangram/gui/Parser.hpp` (l. 104)

```
geometry::Shape operator()(const std::string &path);
```

VII. Chapitre divers : auto, lambda

VII.1. static_assert

Fichier / Numéros de ligne : include/tangram/geometry/Point.hpp (l. 28),

```
static_assert(std::is_arithmetic<T>::value, "Arithmetic type is required");
```

VII.2. Délégation de constructeurs

Fichier / Numéros de ligne : src/tangram/geometry/Polygon.cpp (l. 15-18) Exemple(s) de code

```
Polygon::Polygon(const std::vector<Triangle> &t_triangles, MLV_Color t_color) :
        Polygon(t_color) {
    this->add(t_triangles);
}
```

VII.3. Type alias avec mot-clé using

Fichier / Numéros de ligne : `include/tangram/geometry/Point.hpp` (l. 117)

```
using Vector = Point<T>;
```

VII.4. Inférence de type : mot-clé auto

Fichier / Numéros de ligne : `src/tangram/state/Create.cpp` (l. 54-80)

```
auto saveButton = std::make_shared<gui::ButtonText>(
    BUTTON_X, game::HEIGHT / 2, BUTTON_WIDTH, BUTTON_HEIGHT, 1,
    "Save", "../resources/fonts/helvetica.ttf",
    MLV_rgba(0, 0, 0, 255), MLV_COLOR_BLACK, MLV_COLOR_WHITE,
    MLV_COLOR_GREY70, MLV_COLOR_BLACK, MLV_COLOR_GREY70,
    MLV_COLOR_GREY40, MLV_COLOR_BLACK, MLV_COLOR_GREY40,
    MLV_TEXT_CENTER, MLV_HORIZONTAL_CENTER, MLV_VERTICAL_CENTER,
    &save
);
auto menuButton = std::make_shared<gui::ButtonText>(
    BUTTON_X, game::HEIGHT / 2 + 100, BUTTON_WIDTH, BUTTON_HEIGHT, 1,
    "Menu", "../resources/fonts/helvetica.ttf",
    MLV_rgba(0, 0, 0, 255), MLV_COLOR_BLACK, MLV_COLOR_WHITE,
    MLV_COLOR_GREY70, MLV_COLOR_BLACK, MLV_COLOR_GREY70,
    MLV_COLOR_GREY40, MLV_COLOR_BLACK, MLV_COLOR_GREY40,
    MLV_TEXT_CENTER, MLV_HORIZONTAL_CENTER, MLV_VERTICAL_CENTER,
    [](game::Engine &e) { return e.popState(); }
);
auto quitButton = std::make_shared<gui::ButtonText>(
    BUTTON_X, game::HEIGHT / 2 + 200, BUTTON_WIDTH, BUTTON_HEIGHT, 1,
    "Quit", "../resources/fonts/helvetica.ttf",
    MLV_rgba(0, 0, 0, 255), MLV_COLOR_BLACK, MLV_COLOR_WHITE,
    MLV_COLOR_GREY70, MLV_COLOR_BLACK, MLV_COLOR_GREY70,
    MLV_COLOR_GREY40, MLV_COLOR_BLACK, MLV_COLOR_GREY40,
```

```
    MLV_TEXT_CENTER, MLV_HORIZONTAL_CENTER, MLV_VERTICAL_CENTER,
    [](game::Engine &e) { return e.stop(); }
);
```

VII.5. Itérateur : begin, end

Fichier / Numéros de ligne : src/tangram/geometry/Polygon.cpp (l. 22-25) Exemple(s) de code

```
 std::for_each(
        this->triangles.begin(), this->triangles.end(),
        [&](Triangle &t) { t = t.translate(v); }
);
```

VII.6. Itérateur : cbegin, cend

Fichier / Numéros de ligne : src/tangram/geometry/Polygon.cpp (l. 154-157)

```
std::for_each(
        this->triangles.cbegin(), this->triangles.cend(),
        [this, color](const Triangle &t) { t.rotate(this->angle, this->center).draw(color); }
);
```

VII.7. Boucle foreach

Fichier / Numéros de ligne : src/tangram/geometry/Polygon.cpp (l. 69-78)

```
for (const Triangle &t: this->triangles) {
    auto trianglePoints = t.rotate(this->angle, this->center).getPoints();
    std::for_each(
            trianglePoints.begin(), trianglePoints.end(),
            [&points](const Point16 &p) { points.emplace_back(p); }
    );
}
```

VII.8. Mot-clé explicit pour un constructeur

Fichier / Numéros de ligne : include/tangram/geometry/Polygon.hpp (l. 42) Exemple(s) de code

```
explicit Polygon(MLV_Color color);
```

VII.9. Lambda

Fichier / Numéros de ligne : src/tangram/geometry/Polygon.cpp (l. 22-25) Exemple(s) de code

```
 std::for_each(
        this->triangles.begin(), this->triangles.end(),
        [&](Triangle &t) { t = t.translate(v); }
);
```

VIII. Chapitre constexpr

Fichier / Numéros de ligne : src/tangram/state/Load.cpp (l. 57-62)

```
static constexpr double PREVIEW_SCALE_FACTOR = 0.35;
static constexpr int16_t PREVIEW_SIDE = static_cast<int16_t>(game::HEIGHT * PREVIEW_SCALE_FACTOR);
static constexpr int16_t DEL_BUTTON_SIDE = static_cast<int16_t>(PREVIEW_SIDE * 0.1);
static constexpr int16_t OFFSET_X = (game::WIDTH - PREVIEW_SIDE * 3) / 4;
static constexpr int16_t OFFSET_Y = (game::HEIGHT - PREVIEW_SIDE * 2) / 4;
```

```
static constexpr int16_t BUTTON_WIDTH = PREVIEW_SIDE / 2;
static constexpr int16_t BUTTON_HEIGHT = 30;
```

IX. Chapitre pointeurs intelligents

IV.1. unique_ptr

Fichier / Numéros de ligne : `include/tangram/state/Load.hpp` (l. 38-40)

```
std::unique_ptr<gui::ButtonAbstract> next;
std::unique_ptr<gui::ButtonAbstract> prev;
std::unique_ptr<gui::ButtonAbstract> menu;
```

IV.2. shared_ptr

Fichier / Numéros de ligne : `include/tangram/state/Edit.hpp` (l. 35,36) Exemple(s) de code

```
std::vector<std::shared_ptr<game::Updatable>> updatables = {};
std::vector<std::shared_ptr<gui::Drawable>> drawables = {};
```

X. Chapitre Exceptions

Fichier / Numéros de ligne : `include/tangram/geometry/ParserException.hpp` (l. 21)

```
class ParserException : public std::exception {
```

XI. Chapitre Références universelles, sémantique de déplacement

XI.1. Fonction move

Fichier / Numéros de ligne : `src/tangram/gui/ButtonAbstract.cpp` (l. 7)

```
ButtonAbstract::ButtonAbstract(int16_t t_x, int16_t t_y, int16_t t_w, int16_t t_h,
                               std::function<bool(game::Engine &)> t_execute) :
    x(t_x), y(t_y), w(t_w), h(t_h), execute(std::move(t_execute)) {
}
```

XI.2. Constructeur de déplacement

Fichier / Numéros de ligne : `include/tangram/geometry/Polygon.hpp` (l. 48)

```
Polygon(Polygon &&) = default;
```

XII. ANNEXE UTILISATION DE DESIGN PATTERNS

XII.1. Singleton

Fichier / Numéros de ligne : ‘`include/tangram/state/Load.hpp` (tout le fichier ?)

```
class Play : public ShapeLoaderState {

    private:

            [...]

        Play() = default;


    public:
```

```
        static Play *getInstance();

        [...]
};
```