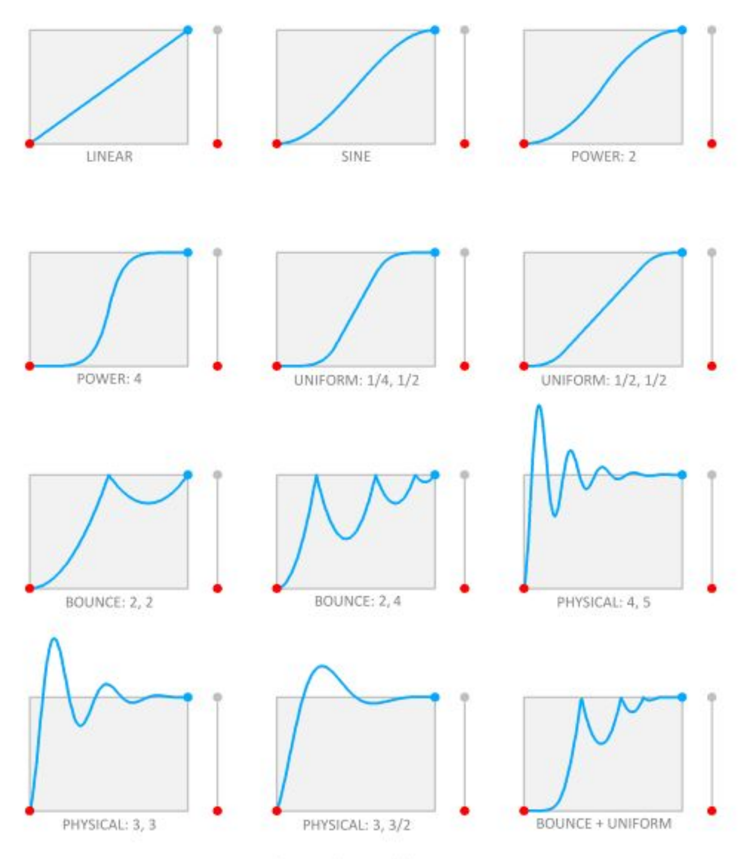
EASING & SPLINES

Christian Piña López
CITM 2019/2020

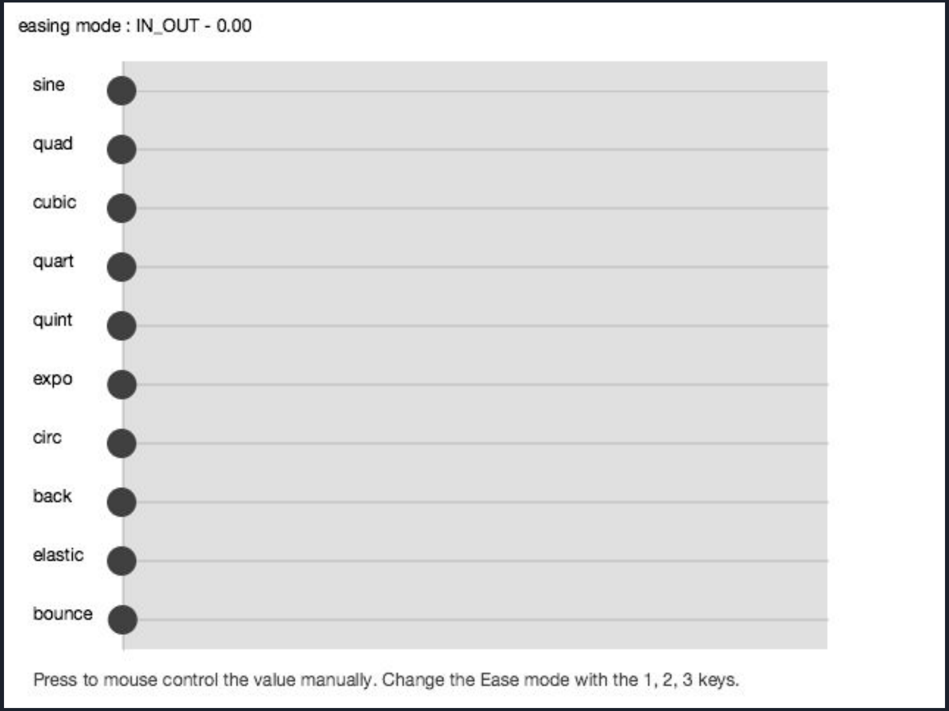
Index

1. What is it and Why is it important?
2. What is a Spline?
3. Spline Usability
4. Easing Functions
5. Splines Types
6. Introduction to the code
7. Todo’s
8. Homework
9. References



1ucasvb.tumblr.com

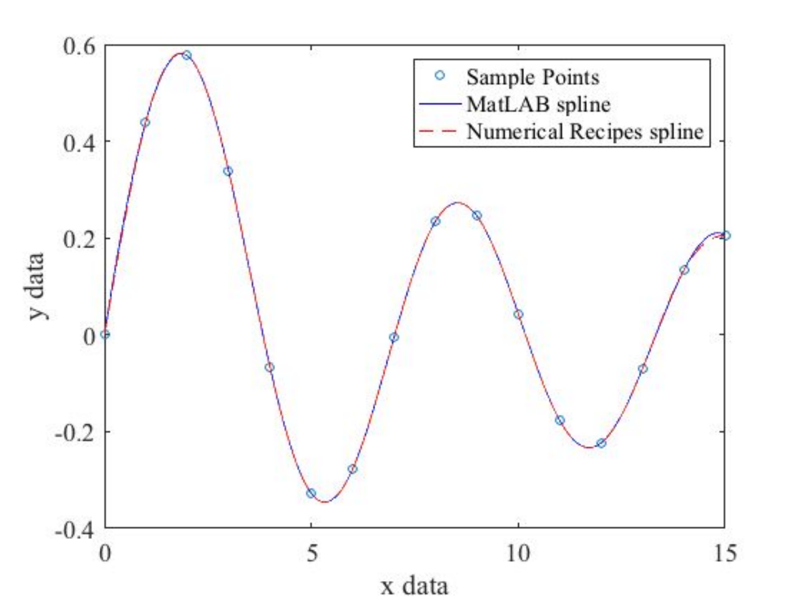
1. What is it and Why is it important?

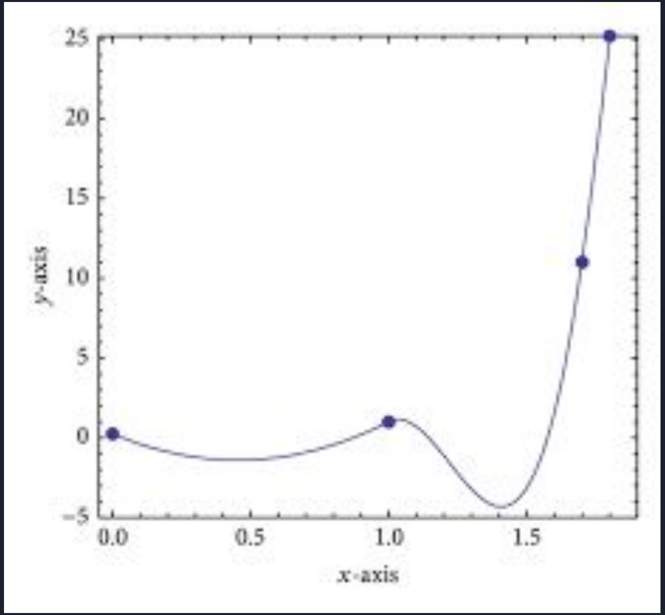


easing mode : IN_OUT - 0.00

Press to mouse control the value manually. Change the Ease mode with the 1, 2, 3 keys.

2. What is a Spline?





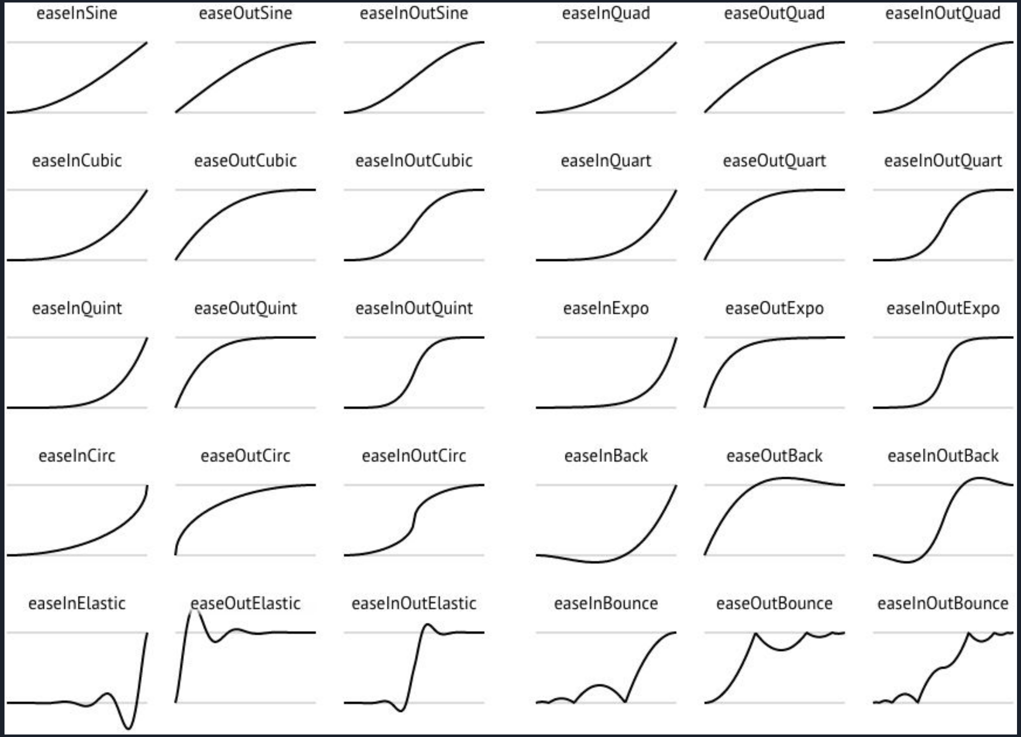
3. Spline Usability

CharacterPath Game Preview Standalone (64-bit/PCD3D_SM5)
File
Digite aqui para pesquisar
Character...
Character...
Sem títul...
14:45
16/05/2017

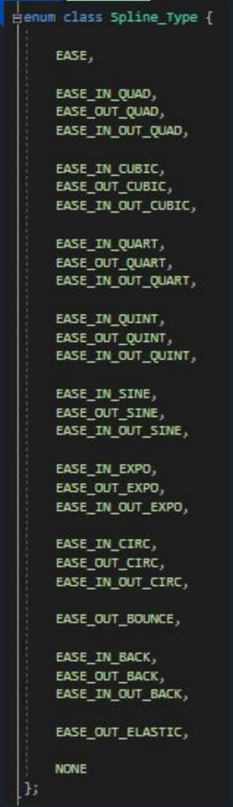
4. Easing Functions

- Initial Value
- Final Value
- Duration
- Total Elapsed Time

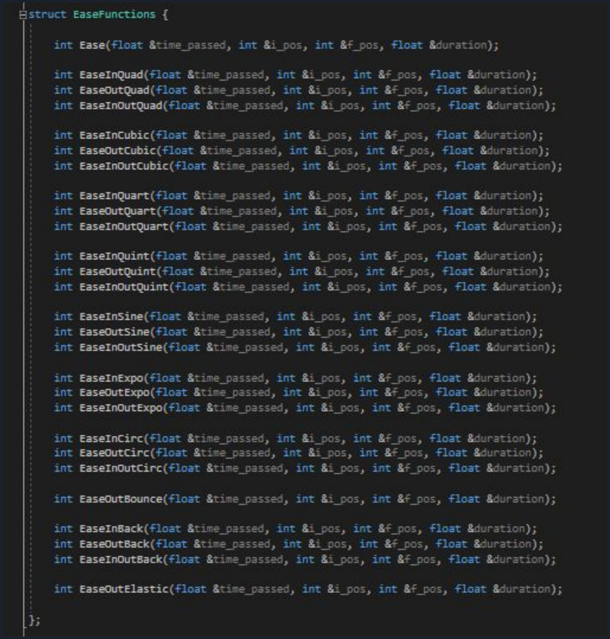
5. Splines Types



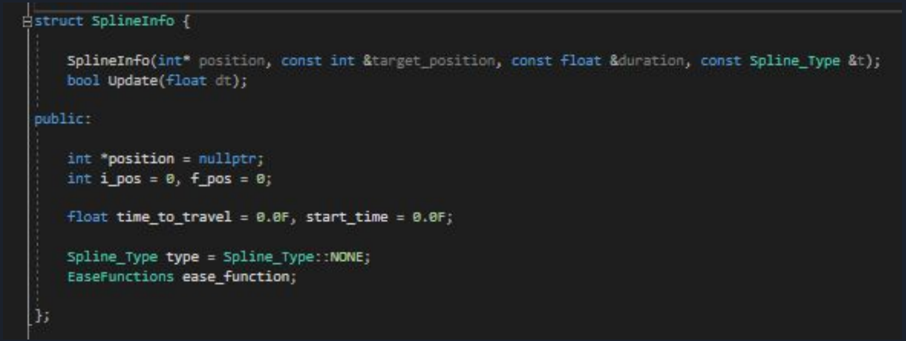
6. Introduction to the code



```
enum class Spline_Type {

    EASE,

    EASE_IN_QUAD,
    EASE_OUT_QUAD,
    EASE_IN_OUT_QUAD,

    EASE_IN_CUBIC,
    EASE_OUT_CUBIC,
    EASE_IN_OUT_CUBIC,

    EASE_IN_QUART,
    EASE_OUT_QUART,
    EASE_IN_OUT_QUART,

    EASE_IN_QUINT,
    EASE_OUT_QUINT,
    EASE_IN_OUT_QUINT,

    EASE_IN_SINE,
    EASE_OUT_SINE,
    EASE_IN_OUT_SINE,

    EASE_IN_EXPO,
    EASE_OUT_EXPO,
    EASE_IN_OUT_EXPO,

    EASE_IN_CIRC,
    EASE_OUT_CIRC,
    EASE_IN_OUT_CIRC,

    EASE_OUT_BOUNCE,

    EASE_IN_BACK,
    EASE_OUT_BACK,
    EASE_IN_OUT_BACK,

    EASE_OUT_ELASTIC,

    NONE
};
```



```
struct EaseFunctions {

    int Ease(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseInQuad(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseOutQuad(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseInOutQuad(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseInCubic(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseOutCubic(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseInOutCubic(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseInQuart(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseOutQuart(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseInOutQuart(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseInQuint(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseOutQuint(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseInOutQuint(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseInSine(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseOutSine(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseInOutSine(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseInExpo(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseOutExpo(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseInOutExpo(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseInCirc(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseOutCirc(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseInOutCirc(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseOutBounce(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseInBack(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseOutBack(float &time_passed, int &i_pos, int &f_pos, float &duration);
    int EaseInOutBack(float &time_passed, int &i_pos, int &f_pos, float &duration);

    int EaseOutElastic(float &time_passed, int &i_pos, int &f_pos, float &duration);

};
```



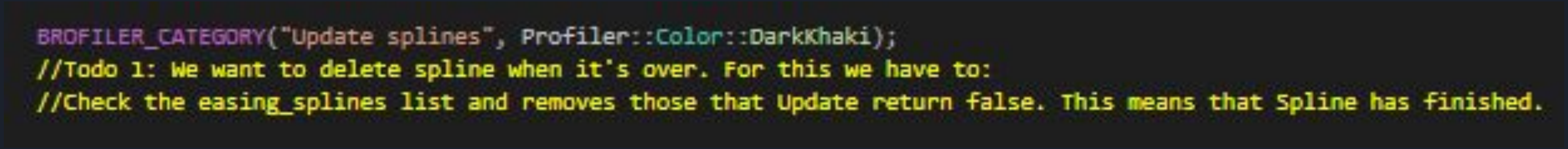
```
struct SplineInfo {

    SplineInfo(int* position, const int &target_position, const float &duration, const Spline_Type &t);
    bool Update(float dt);

public:

    int *position = nullptr;
    int i_pos = 0, f_pos = 0;

    float time_to_travel = 0.0F, start_time = 0.0F;

    Spline_Type type = Spline_Type::NONE;
    EaseFunctions ease_function;

};
```

7. Todo’s

TODO 1

We want to delete spline when it's over. For this we have to:

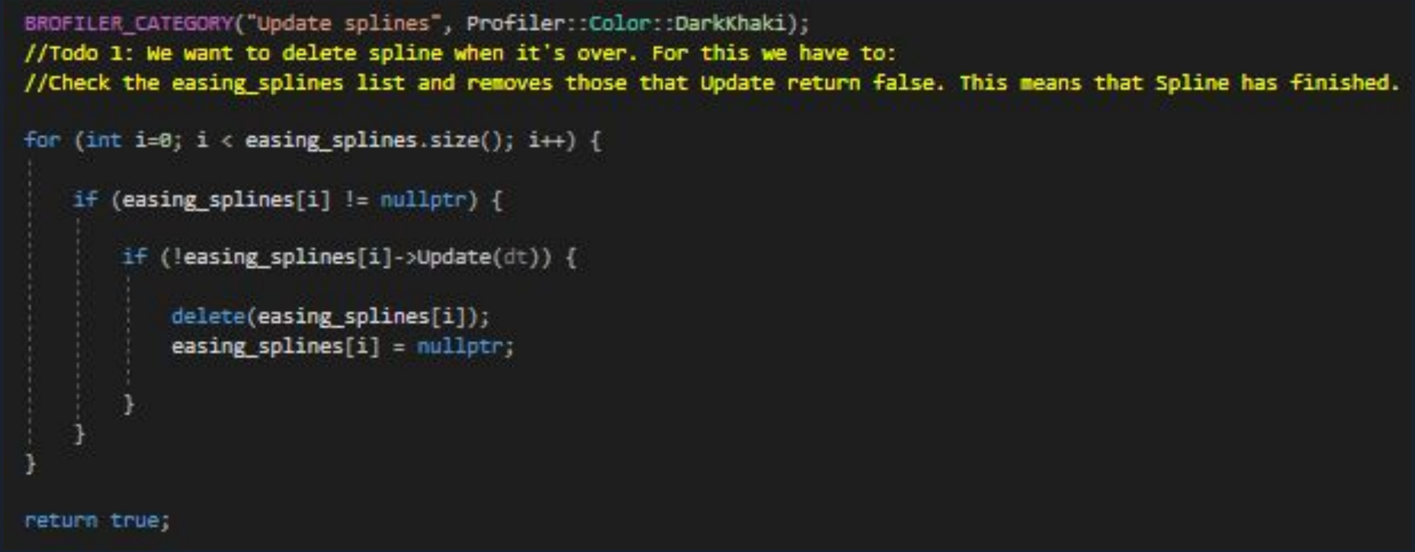
Check the easing_splines list and removes those that Update return false. This means that Spline has finished.



```
BROFILER_CATEGORY("Update splines", Profiler::Color::DarkKhaki);
//Todo 1: We want to delete spline when it's over. For this we have to:
//Check the easing_splines list and removes those that Update return false. This means that Spline has finished.
```

7. Todo’s

TODO 1: Solution

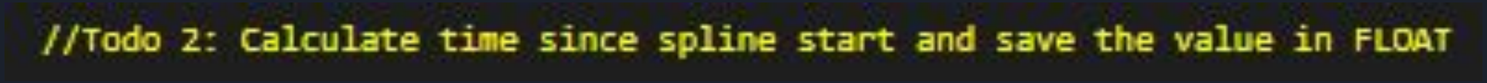


```
BROFILER_CATEGORY("Update splines", Profiler::Color::DarkKhaki);
//Todo 1: We want to delete spline when it's over. For this we have to:
//Check the easing_splines list and removes those that Update return false. This means that Spline has finished.

for (int i=0; i < easing_splines.size(); i++) {

    if (easing_splines[i] != nullptr) {

        if (!easing_splines[i]->Update(dt)) {

            delete(easing_splines[i]);
            easing_splines[i] = nullptr;

        }
    }
}

return true;
```

7. Todo’s

TODO 2

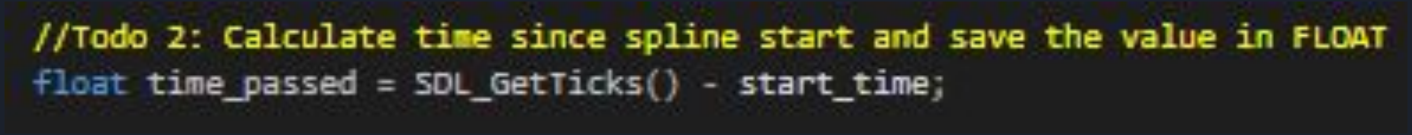
Calculate time since spline start and save the value in FLOAT



```
//Todo 2: Calculate time since spline start and save the value in FLOAT
```

7. Todo’s

TODO 2: Solution

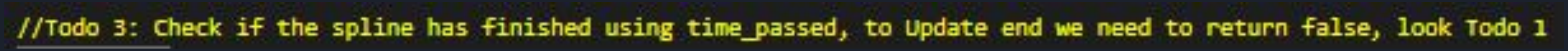


```
//Todo 2: Calculate time since spline start and save the value in FLOAT
float time_passed = SDL_GetTicks() - start_time;
```

7. Todo’s

TODO 3

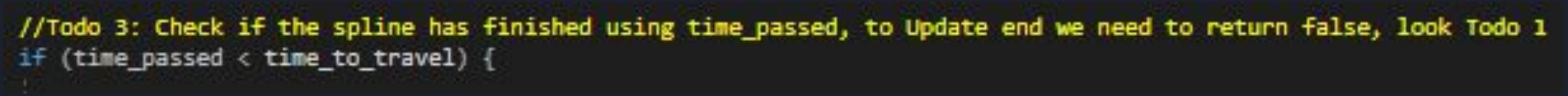
Check if the spline has finished using time_passed, to Update end we need to return false, look Todo 1



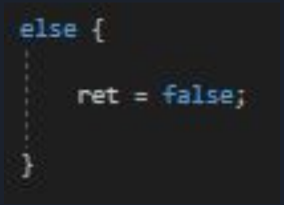
```
//Todo 3: Check if the spline has finished using time_passed, to Update end we need to return false, look Todo 1
```

7. Todo’s

TODO 3: Solution



```
//Todo 3: Check if the spline has finished using time_passed, to Update end we need to return false, look Todo 1
if (time_passed < time_to_travel) {
```

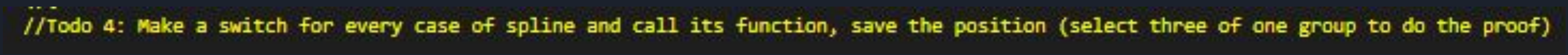


```
else {

    ret = false;

}
```

7. Todo’s

TODO 4

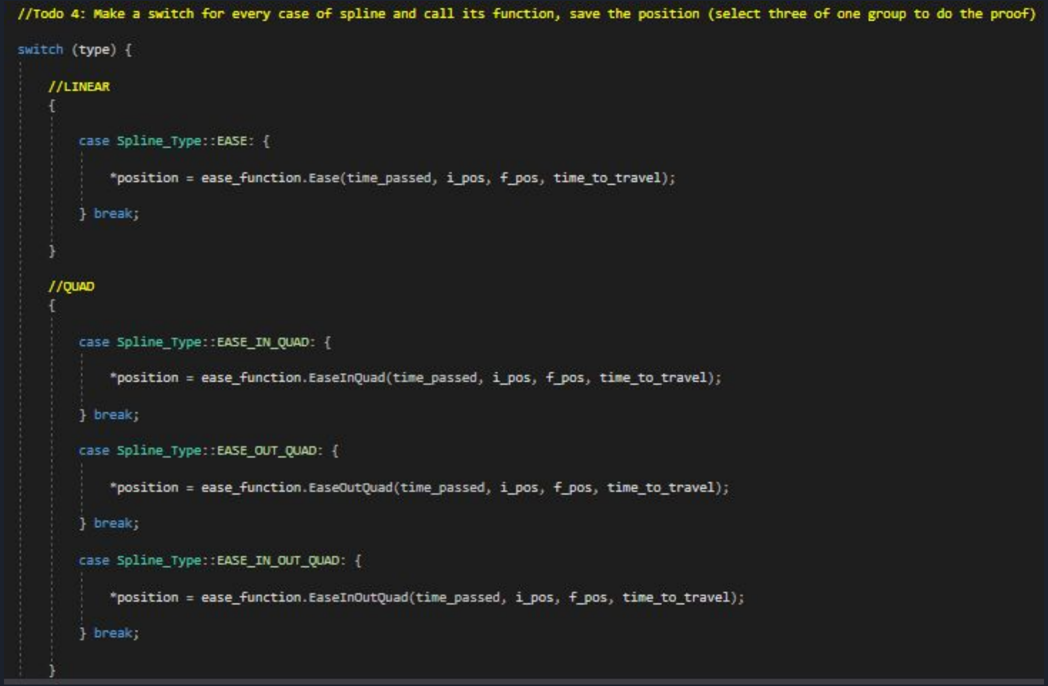
Make a switch for every case of spline and call its function, save the position (select three of one group to do the proof)



```
//Todo 4: Make a switch for every case of spline and call its function, save the position (select three of one group to do the proof)
```

7. Todo’s

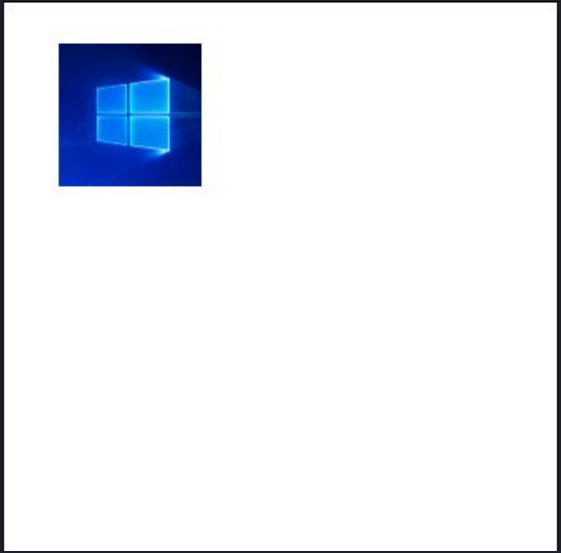
TODO 4: Solution



```
//Todo 4: Make a switch for every case of spline and call its function, save the position (select three of one group to do the proof)

switch (type) {

    //LINEAR
    {

        case Spline_Type::EASE: {

            *position = ease_function.Ease(time_passed, i_pos, f_pos, time_to_travel);

        } break;

    }

    //QUAD
    {

        case Spline_Type::EASE_IN_QUAD: {

            *position = ease_function.EaseInQuad(time_passed, i_pos, f_pos, time_to_travel);

        } break;

        case Spline_Type::EASE_OUT_QUAD: {

            *position = ease_function.EaseOutQuad(time_passed, i_pos, f_pos, time_to_travel);

        } break;

        case Spline_Type::EASE_IN_OUT_QUAD: {

            *position = ease_function.EaseInOutQuad(time_passed, i_pos, f_pos, time_to_travel);

        } break;

    }
```

8. Homework

1. Adapt code to can work with X axis and Y axis simultaneously
2. Adapt code to can work with the scale



9. References

Easing https://github.com/Michaelangel007/easing/

Visual Easing Equations by Robert Penner http://www.gizma.com/easing/#quint1

Creating Usability WIth motion the ux in motion manifesto https://medium.com/ux-in-motion/creating-usability-with-motion-the-ux-in-motion-manifesto-a87a4584ddc

Motion UI IMB Design Lenguage https://www.ibm.com/design/language/motion-ui/basics/

Animating with Robert Penner's Easing Functions https://www.kirupa.com/html5/animating_with_easing_functions_in_javascript.htm

AHEasing Warrenm https://github.com/warrenm/AHEasing