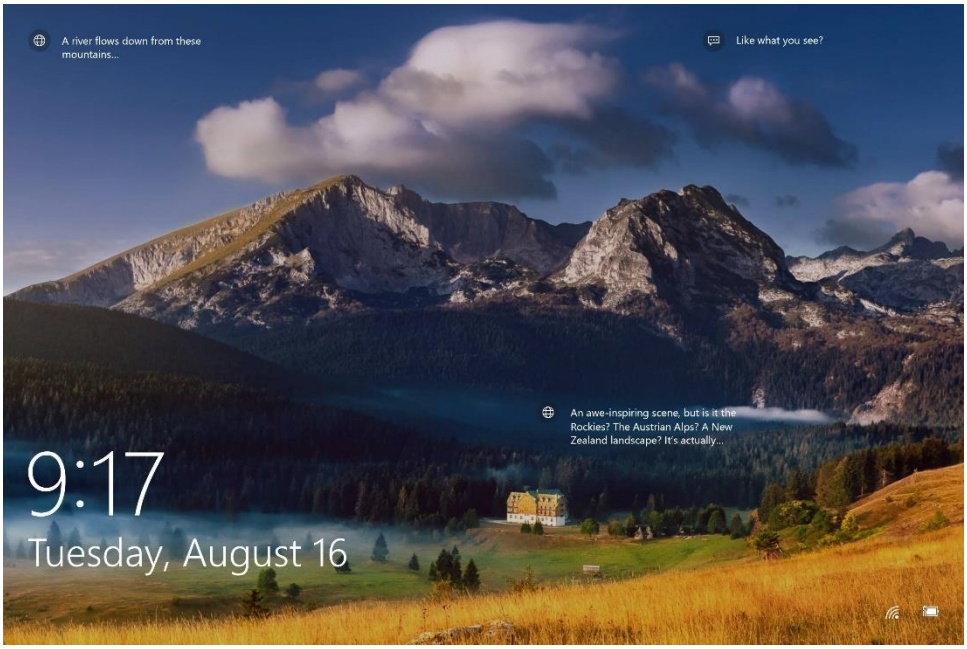
Sugarscreen

Version 2.0.1

DIVERSITY by EPITECH

I. Introduction

Markus Sugarberg, qui s'est fait hacker son jeu récemment, veut renforcer la protection des ordinateurs de son entreprise. Pour ce faire, il a pensé un système super sucrisé qui protège les données, mais les employés se plaignent de l'aspect de l'écran. Aidez-les à égayer leurs journées en créant un super écran de veille.



Exemple d'écran de veille

.

II. Consignes

- Lisez tout avant de commencer !
- Pour ce projet, il vous sera demandé de choisir comme nom de repository : cc_sugarscreen
- Si tu bloques, rappelles-toi que tu es accompagné(e) ! Demande de l'aide à tes camarades ou à un Cobra. L'union fait la force !
- Internet est un outil formidable pour découvrir le fonctionnement des choses, Google est ton ami sers-t 'en régulièrement !
- Vous pouvez utiliser la documentation de processing pour en apprendre plus : Processing

Explication du code :

- Un `...` dans le code signifie que tu dois compléter le code par toi-même en utilisant les informations du sujet.
- Tout ce qui est écrit après un `//` est un commentaire pour t'aider à comprendre.
- Les exemples de code fourni ci-dessous sont incomplets, vous devrez rajouter ou modifier quelques éléments pour que cela fonctionne.

III. Les bases du glucose

Pour commencer cette mission, essayez de reproduire ce fond d'écran très connu avec un objet qui rebondit contre les bords de l'écran comme ici. La première étape, est d'afficher des cercles afin dès les faire bouger. Cela nécessite la mise en place de certaine fonctions : setup() et draw(). Elles sont obligatoires pour le fonctionnement du programme en Processing. Voici un début de code pour obtenir un cercle :

```
void setup() {
    size(600, 600); //définit la taille de la fenêtre
    background(100); //définit la couleur de fond de la fenêtre, 100
= gris
    fill(255, 0, 0); // remplit le cercle en rouge
    ellipse(width/2, height/2, 50, 50);
}

void draw(){
    fill(0, 0, 255); //remplit le cercle en bleu
    ellipse(width/2, height/2, 50, 50);
}
```

On constate que le cercle au centre est bleu. C'est parce que la fonction draw() est appelée après setup(). Comme son nom l'indique, setup() sert à préparer le reste du programme. La fonction passera donc une seule fois par cette fonction au tout début.

A l'inverse draw(), se traduit par dessin, et est appelé plusieurs fois pour réafficher à chaque fois ce qu'on lui demande. Ajoutez votre grain de sucre... Essayez de faire bouger quelque chose maintenant ! Pour avoir un peu plus de mouvement, il va falloir ajouter une variable “y” qui sera un entier et augmentera de 1 à chaque fois que l’on passe par draw(), puis utilisez le dans la fonction ellipse().

Ensuite, pourquoi pas ajouter un background pour avoir un beau fond !

 Bravo ! Vous avez atteint le niveau 1 ! 

IV. Une super lancé de sucre !

Désormais il faut stocker les coordonnées du cercle, pour cela utilisez PVector. Premièrement donnez un nom à “PVector pos”, après initialisez-le à 0.

```
PVector pos = new PVector(0, 0);
```

Parfait, vous avez un bonbon rond qui bouge ! Cependant, ça bouge beaucoup et malheureusement il part de l’écran... Essayez de le faire rester dans l’écran. Pour y arriver il va falloir se poser deux questions :

* A quel moment rebondir ?
* Comment rebondir ?

La première partie se compose d’une condition pour chaque bord de la fenêtre, cherchez à y faire une action, seulement si un bord est atteint. En l'occurrence les conditions se résume ainsi : Si la balle passe le bord de la fenêtre alors il va falloir inverser le sens de déplacement du cercle.

Petit conseil : Si vous augmentez de 1 le cercle pour le faire aller vers le bas de combien doit-on augmenter pour qu'il remonte ?

Ajoutez une petite touche de salé, faites-en sorte que **:**

* Votre cercle se déplace en x et en y
* Votre cercle rebondisse sur toutes les parois de la fenêtre
* Votre cercle commence avec des vitesses aléatoires en x et en y

L'aléatoire c'est une base en programmation il y a sans doute quelque chose pour ça... Bien évidemment, vous n’êtes pas obligée de le faire maintenant, mais garder cette idée en tête !

Prenez un bonbon pour vous récompenser !

Félicitations !

V. Les sucres en confinement

Jusqu'ici vous avez utilisé une méthode appelée "itératif". Vous allez maintenant s'intéresser aux classes et la "programmation orientée objet". Vous allez donc commencer par créer une class. Une classe, c'est ce qui vous permet de créer des objets. Imaginez-la “class” comme le moule avec lequel on fait un bonbon.

```
class Ball {
    // les objets créés avec cette class auront toutes ces "variables
membres"
    PVector pos;
    int size;


    Ball(int _size){
        pos = new PVector(width/2, height/2);
        size = _size;
    }
    //display doit afficher un cercle aux coordonnées pos de taille
size par size
    void display(){
        ...
    }
}
```

VI. Faire un bonbon c’est top, mais maintenant que faire ?

Qu’est-ce qu’il est bon ce bonbon ! Mais un seul c’est ennuyeux … Il faut donc en recréer d’autres… Créez-en une armée entière, grâce à l’ingrédient : new.

```
Ball ball = new Ball(50) ;
```

Maintenant l'objet peut modifier ses propres variables et se débrouiller tout seul. Il a juste besoin qu'on lui dise quand le faire. On peut donc appeler ses fonctions membres comme ci-dessous :

```
ball.display() ;
```

Let’s go, ça va monter d’un cran !

* Réécrire les fonctions setup() et draw() pour afficher votre cercle
* Écrire une fonction membre à Ball qui la fait bouger
* Écrire une fonction stayIn() qui empêche la balle de sortir de la fenêtre

Encore un effort ! Vous y êtes presque !

VII. L'armée de bonbon à l'attaque !

Avoir un bonbon c'est bien, mais en avoir une armée de bonbons c'est mieux !

```
Ball balls[] = new Ball[100];

for(int i = 0;  i < balls.length; i++) {
   balls[i] = new Ball(10);
}
for(int i = 0; i < balls.length; i++) {
    balls[i].display();
    … // appel les autres fonctions de Ball
}
```

Et voilà pour les classes ! Bravo ! Vous avez maintenant quelque chose qui ressemble à un fond d'écran de veille Windows vintage !

VIII. Vers le caramel et au-delà

Voilà, votre Sugarscreen est fonctionnel, maintenant c'est à vous de rajouter la cerise sur le gâteau ! Si vous vous en sentez le courage, voici des axes d'améliorations possibles.

* Ajouter une couleur aléatoire à la création d'un bonbon
* Ajoutez des images de bonbons à la place des boules
* Ajoutez une image de fond (chocolaterie par exemple)
* Réalisez des mouvements aléatoires sur les bonbons
* Rendre les bonbons cliquables pour les collecter, incrémentez son score, changez de couleur les bonbons etc ...
* Ajoutez des bonbons en appuyant sur une touche de votre clavier

Prenez un bonbon au passage !