Codejam Space Program

MANUEL D'EXPLICATIONS

Dans ce document, vous trouverez les explications derrière notre programme, de l'affichage à la physique qui permet à la fusée de fonctionner.

Pour plus de détails, veuillez vous référer aux sources, qui seront fournies avec ces documents.

Bonne lecture!

Signé: *Les noobs du Cegep de Jonquière 2*

Affichage

Au tout début du programme, la console s'ajuste à une taille standard de portable, afin de permettre à la plupart des ordinateurs ayant un petit écran d'être compatibles avec les graphiques du programme.

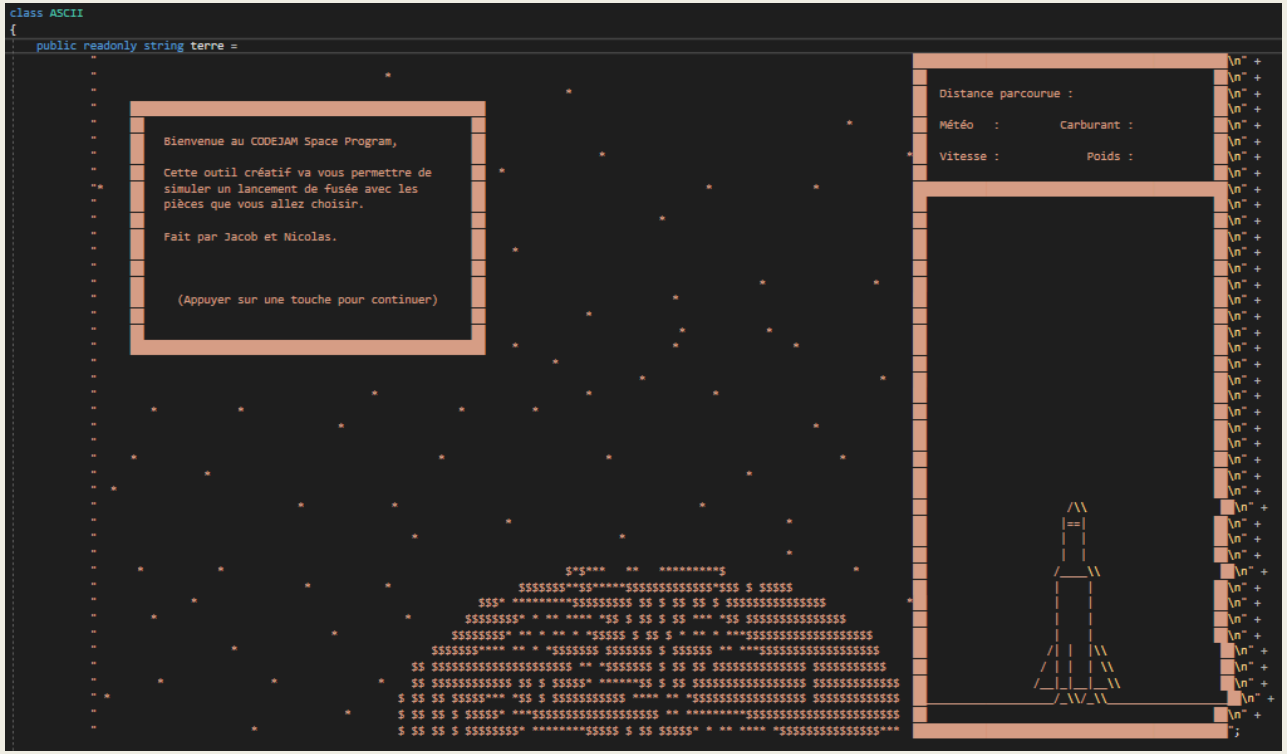
Les boîtes de dialogues ne sont jamais supprimées, mais bien remplacées par d'autres boîtes, ou bien une boîte vide, afin de maximiser la beauté des animations.

```
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace CodeJam_SPACE
8   {
9       class Affichage
10      {
11          private ASCII ascii = new ASCII();
12          private Random random = new Random();
13          public void init()
14          {
15              Console.SetWindowSize(170, 44);
16              Console.WriteLine(ascii.terre);
17              Console.SetCursorPosition(51, 15);
18          }
19          public void effacerTextBox()
20          {
21              Console.SetCursorPosition(7, 4);
22              Console.WriteLine("                                                  ");
23              Console.SetCursorPosition(7, 5);
24              Console.WriteLine("                                                  ");
25              Console.SetCursorPosition(7, 6);
26              Console.WriteLine("                                                  ");
27              Console.SetCursorPosition(7, 7);
28              Console.WriteLine("                                                  ");
29              Console.SetCursorPosition(7, 8);
30              Console.WriteLine("                                                  ");
31              Console.SetCursorPosition(7, 9);
32              Console.WriteLine("                                                  ");
33              Console.SetCursorPosition(7, 10);
34              Console.WriteLine("                                                  ");
35              Console.SetCursorPosition(7, 11);
36              Console.WriteLine("                                                  ");
37              Console.SetCursorPosition(7, 12);
38              Console.WriteLine("                                                  ");
39              Console.SetCursorPosition(7, 13);
40              Console.WriteLine("                                                  ");
41              Console.SetCursorPosition(7, 14);
42              Console.WriteLine("                                                  ");
43              Console.SetCursorPosition(7, 15);
44              Console.WriteLine("                                                  ");
45              Console.SetCursorPosition(7, 16);
46              Console.WriteLine("                                                  ");
47              Console.SetCursorPosition(7, 17);
48              Console.WriteLine("                                                  ");
49          }
50          public void tuto()
51          {
```

Affichage (continuation)

Pour ce qui en est des graphiques, la plupart de l'affichage est fait à l'aide de `Console.Write()`, avec une combinaison de texte et de dessins ASCII.

Tous les dessins fait en ASCII ont été fait à la main, avec le programme de bloc-note.



Physique

Le cœur du programme, le système de physique dicte le fonctionnement de la fusée, puis renvoie automatiquement les données vers l'affichage.

Les formules utilisées sont tirées de divers sites, notamment le site de la NASA. Malheureusement, la résistance de l'air n'est pas pris en compte, puisqu'elle est déterminée expérimentalement. De plus, les formules sont simplifiées pour une compréhension plus facile.

```
class Physique
{
    /*Décolage:
     *          https://tpefusee.wordpress.com/previsions-de-la-mission-les-parametre-de-lancement/
     * 1 tonne = 1000kg
     */

    private readonly double INTENSITE_GRAV_TERRE = 9.8/*N/kg*/;
    private double poidsFusee, masseFusee, pousseeFusee, debitMasique, accelerationFusee, impulsionSpecifique;
    private Fusee fusee;
    private Affichage affichage = new Affichage();


    public Physique(Fusee fusee)
    {
        this.fusee = fusee;
        //sans carburant
        masseFusee = fusee.getPoidsTotal();
        pousseeFusee = fusee.thrust();
        impulsionSpecifique = fusee.impulsionSpecifique();
        QuantiteCarburant = fusee.getQuantiteCarburant();
    }
    void CalculerPoidsFusee()
    {
        poidsFusee = (masseFusee + QuantiteCarburant) * INTENSITE_GRAV_TERRE;
    }
    void CalculerAcceleration()
    {
        accelerationFusee = (pousseeFusee - poidsFusee) / (masseFusee + QuantiteCarburant); //Voir http://www.rocket-simu
    }
    void CalculerPerteMasse()
    {
        debitMasique = pousseeFusee / impulsionSpecifique;
    }
    void CalculerVitesseFusee()
    {
        VitesseFusee += accelerationFusee;
        if (VitesseFusee < 0)
            VitesseFusee = 0;
    }
    void CalculerHauteurFusee()
    {
        Hauteur += VitesseFusee;
    }
    public void MiseAJour()
    {
        int timer = 0;
        VitesseFusee = 1;
        while (VitesseFusee > 0)
        {
            System.Threading.Thread.Sleep(250);
```

Pièces

La classe `Fusée.cs` est composée de différentes classes, qui agissent comme étant les pièces de la fusée.

Lorsqu'une pièce est choisie par l'utilisateur, celle-ci transmet ses informations à la classe `Physique.cs`, qui s'occupera de déterminer automatiquement la trajectoire, la consommation de carburant et le poids total de la fusée.

```
class Fusee
{
    private Cabine cabine;
    private Moteur moteur;
    private Carburant carburant;
    public Fusee(Cabine cabine, Moteur moteur, Carburant carburant)
    {
        this.cabine = cabine;
        this.moteur = moteur;
        this.carburant = carburant;
    }
    public double getPoidsTotal()
    {
        return cabine.Poids + moteur.Poids + 500 /*Poids du réservoir*/;
    }
    public double thrust()
    {
        return moteur.Thrust;
    }
    public double impulsionSpecifique()
    {
        return moteur.ImpSpecifique;
    }
    public double getQuantiteCarburant()
    {
        return carburant.Quantite;
    }
```

Station

La classe `Station.cs` s'occupe de la gestion de l'affichage, la physique et le montage de la fusée. Les choix des utilisateurs est fait dans cette classe, à l'aide de la fonction suivante: `FaireChoix(ref int variable)`, qui permet des choix sans que le programme ne plante lors d'une erreur de frappe.

La fusée est par la suite lancée, suite aux choix.

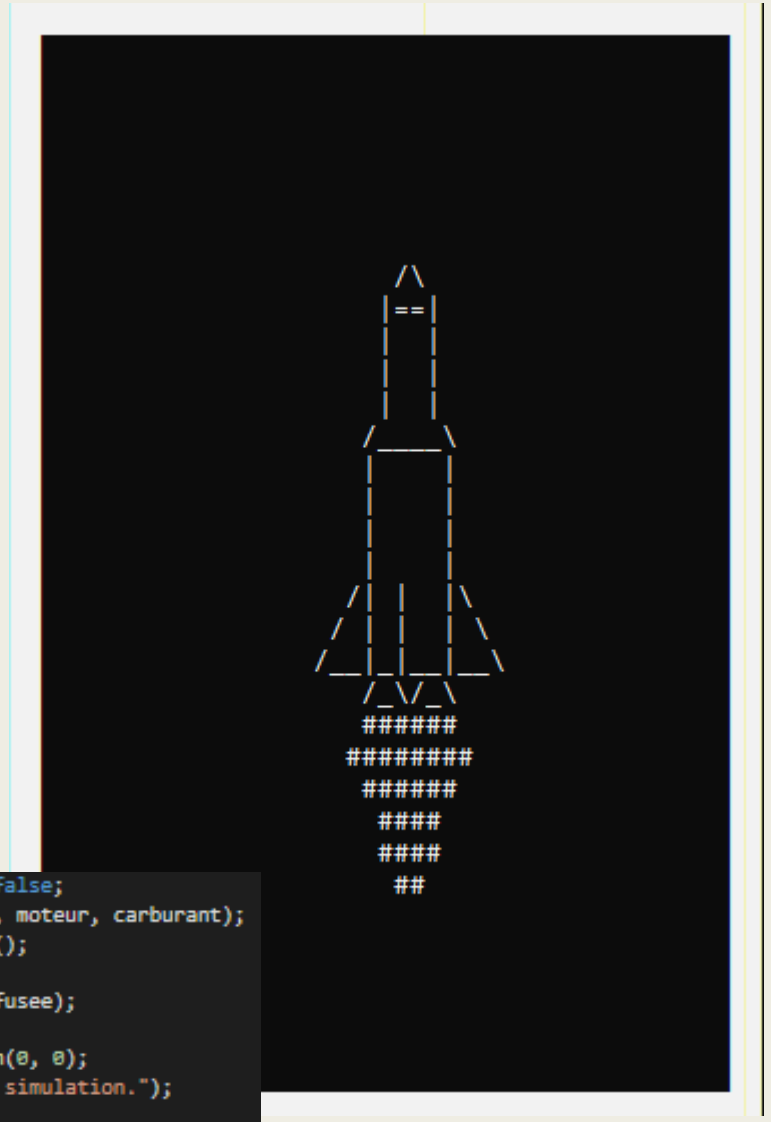
```
class Station
{
    private MeteoActuel meteo;
    private Affichage affichage = new Affichage();
    private List<Meteo> meteos = new List<Meteo>();
    private Cabine cabine;
    private Moteur moteur;
    private Carburant carburant;
    private Fusee fusee;
    private Physique physique;
    private int choix;
    private int quantiteCarburant;

    public void FaireChoix(ref int variable)
    {
        try
        {
        variable = Convert.ToInt16(Console.ReadLine());
        }
        catch (Exception) {}
    }
    public void play()
    {
        //Instancier la météo
        meteo = new MeteoActuel(meteos);
        //Donner le choix de la Cabine
        do
        {
            affichage.choixCockpit();
            FaireChoix(ref choix);
        }
        while (choix != 1 && choix != 2 && choix != 3);
        switch (choix)
        {
            case 1:
                cabine = new Cabine("Light Cockpit MK2", 832);
                break;
            case 2:
                cabine = new Cabine("Dio Cockpito", 1071);
                break;
            case 3:
                cabine = new Cabine("CNSA Cockpit", 1674);
                break;
        }
        //Donner le choix du Moteur
        do
        {
            affichage.choixEngine();
            FaireChoix(ref choix);
```

Lancement de la fusée

Finalement, lorsque les pièces sont choisies, plusieurs opérations sont effectuées:

-La physique est calculée automatiquement selon les pièces choisie.
-La boîte de dialogue est supprimée.
-Une boucle fait fonctionner la simulation jusqu'à ce que la vitesse de la fusée soit de zéro; la simulation se termine, car à ce point, la fusée commencerait à redescendre sur Terre.

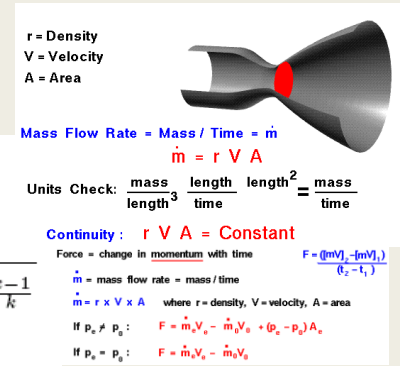


```
Console.CursorVisible = false;
fusee = new Fusee(cabine, moteur, carburant);
affichage.effacerTextBox();
affichage.Lancement();
physique = new Physique(fusee);
physique.MiseAJour();
Console.SetCursorPosition(0, 0);
Console.Write("Fin de la simulation.");
Console.ReadKey();
```

Un commentaire des noobs...

Nous vous remercions pour ce concours unique, qui nous a permis de nous développer en tant que programmeur et artiste, tout en nous amusant! Les difficultés furent complexes, mais à la fin, nous sommes fiers de notre réussite!

Quelques statistiques amusantes...

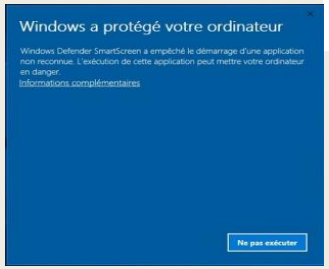


$$\Delta v = v_e \ln \frac{m_0}{m_f} = I_{sp} g_0 \ln \frac{m_0}{m_f}$$

$$V_e = \sqrt{\left(\frac{2k}{k-1}\right)\left(\frac{RT_c}{M}\right)\left(1 - \frac{P_e}{P_c}\right)^{\frac{k-1}{k}}}$$

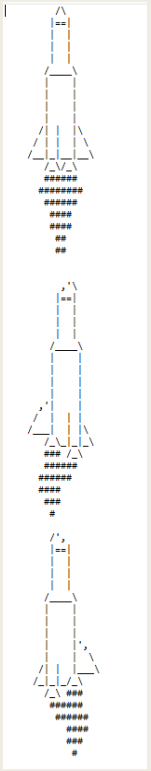
-Heures passées sur le site de la NASA: 10 heures+

-Nombre de reprises de la vidéo: ~18 fois:



-Nombre de fois que Jacob fut… audiblement malade: trop

-Heures passées sur bloc-note(dessin ASCII): 3 heures+



-Nombre de lignes de code:

Final: 993 lignes | Estimé total: 1300 lignes

```
/*double CalculerVitesseEjectionGaz()
{
    //Ve = Rac((2k/k-1)(RTc/M)(1-Pe/Pc)^(k-1/k)) || Pe = 1bar, M = déclaré avec carburant, Pc = déclaré avec chambre combustion(moteur), Tc = déclaré avec carburant, R = 8.31, K = Gamma(voir https://fr.wikipedia.org/wiki/Indice_adiabatique)
    vitesseEjectionGaz = Math.Sqrt(Math.Pow((((2*gammaGaz)/(gammaGaz-1))*((CONSTANTE_GAZ_PARFAITS - 1)/CONSTANTE_GAZ_PARFAITS)*(1 - (pressionExterieure/pressionCombustion))),((gammaGaz - 1)/gammaGaz)));
    return vitesseEjectionGaz; //m/s
}
double CalCulerVitesseDecollage()
{
    //F = Dm * Ve|| F = poussée, Dm= débit massique de gaz éjecté(déclaré avec moteur)
    pousseeFusee = debitMassiqueGaz * vitesseEjectionGaz;
    return pousseeFusee;
}*/
```