Approximation de π à la Monte Carlo avec geogebra

Groupe mathématiques dynamiques – Irem de Lyon

1 Objectifs du document

1. Illustrer une méthode algorithmique classique.
2. Éléments de formation sur geogebra : scripts avec du python, scripts avec du javascript.

2 Préparation de la « toile de fond »

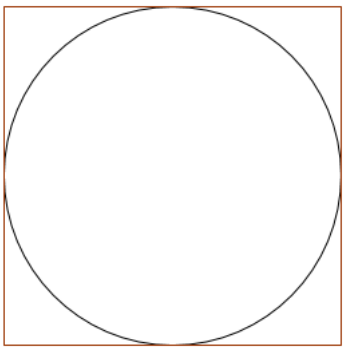
On commence par définir le carré de sommets $A(-1\ ;\ -1)$, $B(1\ ;\ -1)$, $C(1\ ;\ 1)$, $D(-1\ ;\ 1)$:

Saisie: **Polygone[(-1,-1),(1,-1),(1,1),(-1,1)]**

puis le cercle de centre $O(0\ ;\ 0)$ et de rayon 1 :

Saisie: **Cercle[(0,0),1]**

Avec un clic droit sur le carré, sélectionner « propriétés » et rendre le fond transparent :
On obtient cette magnifique figure :

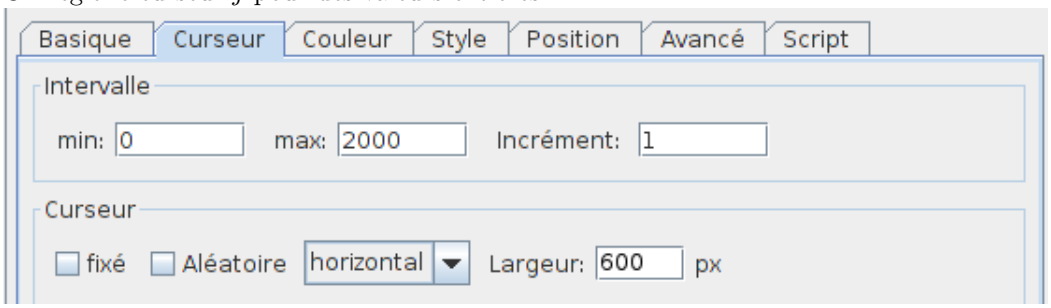


On définit également un curseur j (que l'on affichera dans l'écran graphique comme curseur par un clic droit sur l'objet) et un curseur $nbPointsIn$.

Saisie: **j=1**

Saisie: **nbPointsIn=1**

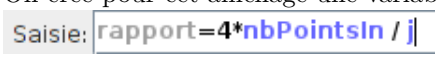
On règle le curseur j pour des valeurs entières :



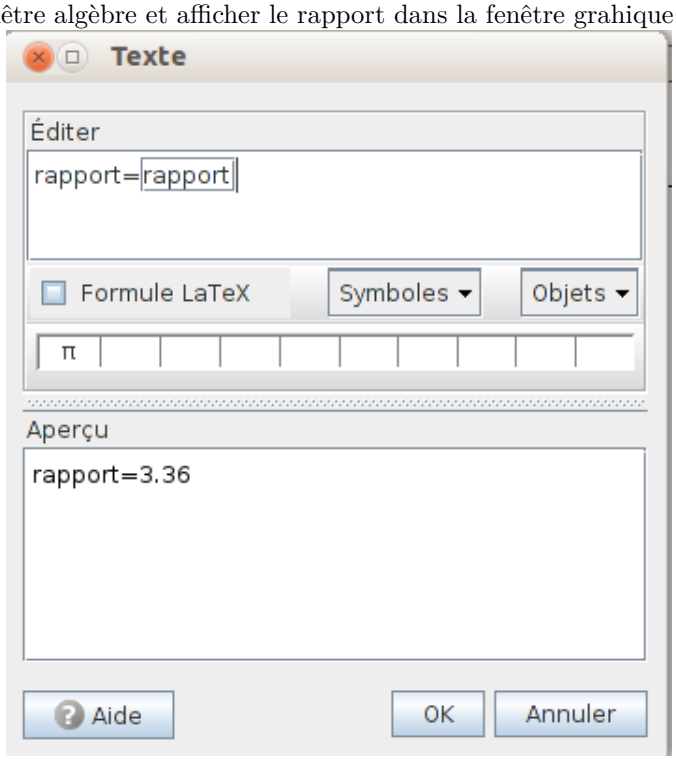
L'objectif est ensuite de tirer j points au hasard dans le carré et de compter le nombre $nbPointsIn$ de ces points qui se trouvent dans le disque.

L'affichage du rapport $\frac{4 \times nbPointsIn}{j}$ semble approcher un nombre bien connu…

On crée pour cet affichage une variable rapport :



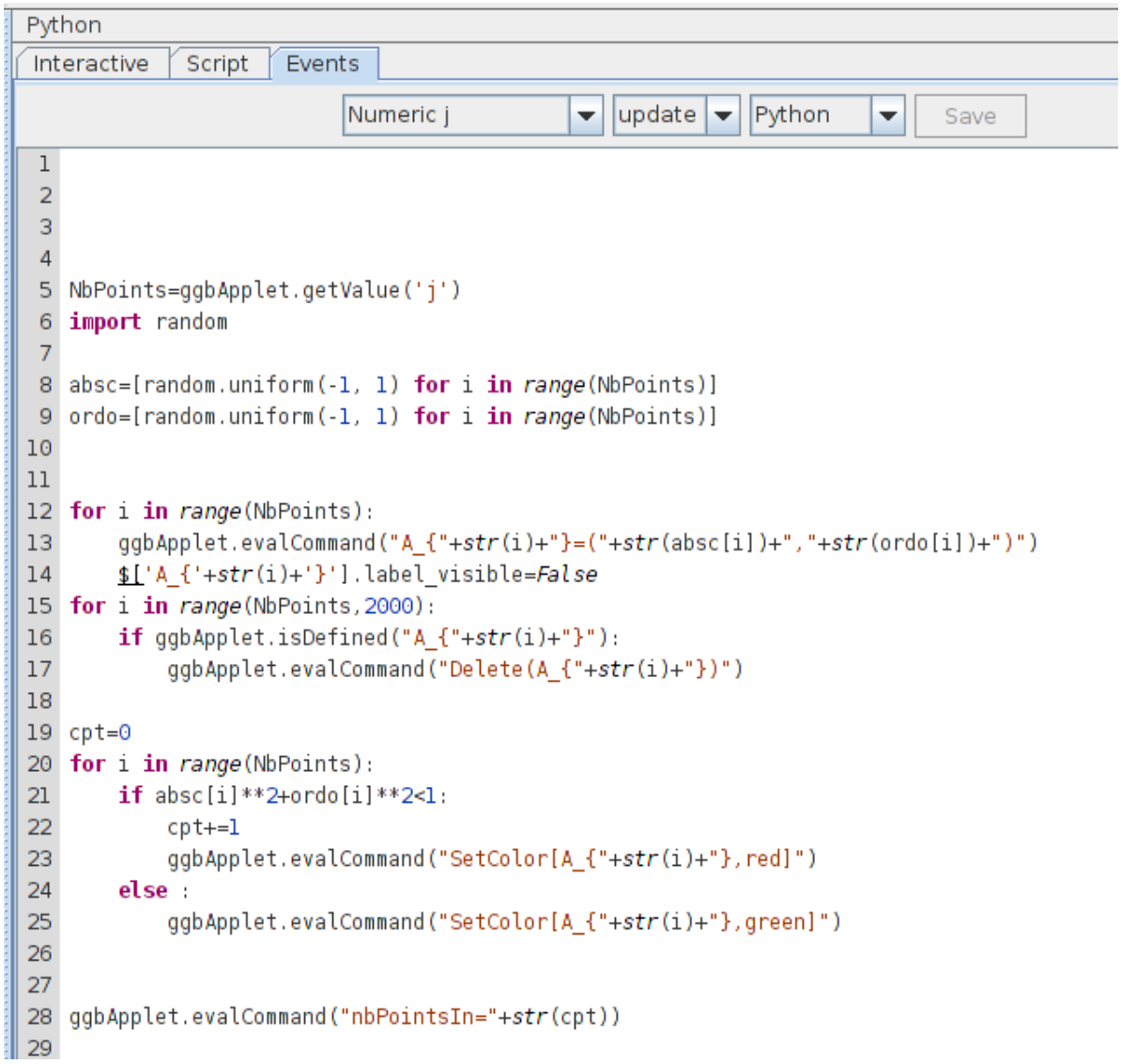
La fenêtre d'algèbre étant vite alourdie par la création de nombreux points dans la suite, on peut fermer cette fenêtre algèbre et afficher le rapport dans la fenêtre grahique à l'aide de l'outil « insérer texte » :



Le premier mot « rapport » est un simple texte (tapé au clavier), le second (encadré) est obtenu à partir du menu déroulant « objets » et affichera la valeur de la variable gg rapport.

3 Avec python

On associe maintenant un script python au curseur j (onglet events de la fenêtre python).



Script 1 – PythonGG

```
NbPoints=ggbApplet.getValue('j')
import random

absc=[random.uniform(-1, 1) for i in range(NbPoints)]
ordo=[random.uniform(-1, 1) for i in range(NbPoints)]


for i in range(NbPoints):
        ggbApplet.evalCommand("A_{"+str(i)+"}=("+str(absc[i])+","+str(ordo[i])+")")
        $['A_{'+str(i)+'}'].label_visible=False
for i in range(NbPoints,2000):
        if ggbApplet.isDefined("A_{"+str(i)+"}"):
                ggbApplet.evalCommand("Delete(A_{"+str(i)+"})")

cpt=0
for i in range(NbPoints):
        if absc[i]**2+ordo[i]**2<1:
                cpt+=1
                ggbApplet.evalCommand("SetColor[A_{"+str(i)+"},red]")
```

```
        else :
                ggbApplet.evalCommand("SetColor[A_{"+str(i)+"},green]")


ggbApplet.evalCommand("nbPointsIn="+str(cpt))
```

Décryptons un peu ce code :

Script 2 – PythonGG

```
NbPoints=ggbApplet.getValue('j')
```

NbPoints est une variable python dans laquelle on stocke la valeur actuelle du curseur j.

Script 3 – PythonGG

```
import random

absc=[random.uniform(-1, 1) for i in range(NbPoints)]
ordo=[random.uniform(-1, 1) for i in range(NbPoints)]
```

On importe la bibliothèque random de python et on tire au hasard deux listes de taille NbPoints constituées de nombres tirés au hasard, suivant la loi uniforme, dans l'intervalle $[-1\ ;\ 1]$.

Script 4 – PythonGG

```
for i in range(NbPoints):
        ggbApplet.evalCommand("A_{"+str(i)+"}=("+str(absc[i])+","+str(ordo[i])+")")
        $['A_{'+str(i)+'}'].label_visible=False
for i in range(NbPoints,2000):
        if ggbApplet.isDefined("A_{"+str(i)+"}"):
                ggbApplet.evalCommand("Delete(A_{"+str(i)+"})")
```

On définit les points A_0, A_1, ..., $A_{\text{NbPoints}-1}$, l'abscisse de A_i étant l'élément d'indice i de la liste absc et l'ordonnée de ce même point l'élement d'indice i de la liste ordo.

On demande ensuite que les étiquettes des points ne soient pas visibles dans la fenêtre graphique de geogebra.

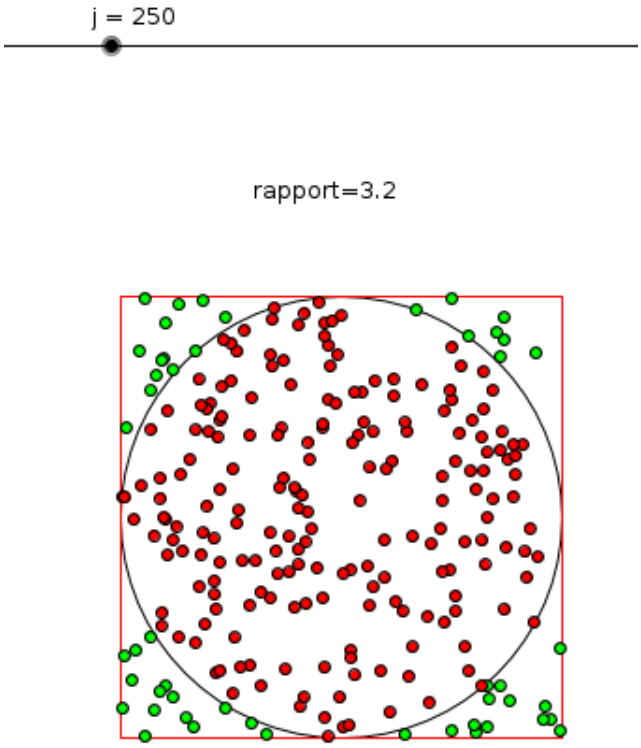
On détruit enfin tous les points dont l'indice est $\geqslant$ NbPoints (on a fixé arbitrairement le nombre maximal de points à 2000 lors de la création du curseur j).

Script 5 – PythonGG

```
cpt=0
for i in range(NbPoints):
        if absc[i]**2+ordo[i]**2<1:
                cpt+=1
                ggbApplet.evalCommand("SetColor[A_{"+str(i)+"},red]")
        else :
                ggbApplet.evalCommand("SetColor[A_{"+str(i)+"},green]")


ggbApplet.evalCommand("nbPointsIn="+str(cpt))
```

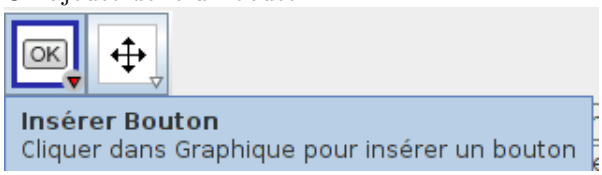
Dans cette partie, on compte le nombre de points à l'intérieur du cercle (dans ce cas, le point est affiché en rouge).
Puis la variable geogebra nbPointsIn est mise à jour par l'instruction ggbApplet.evalCommand("nbPointsIn="+str(cpt)).

En tirant le curseur j vers la droite, on obtient par exemple :

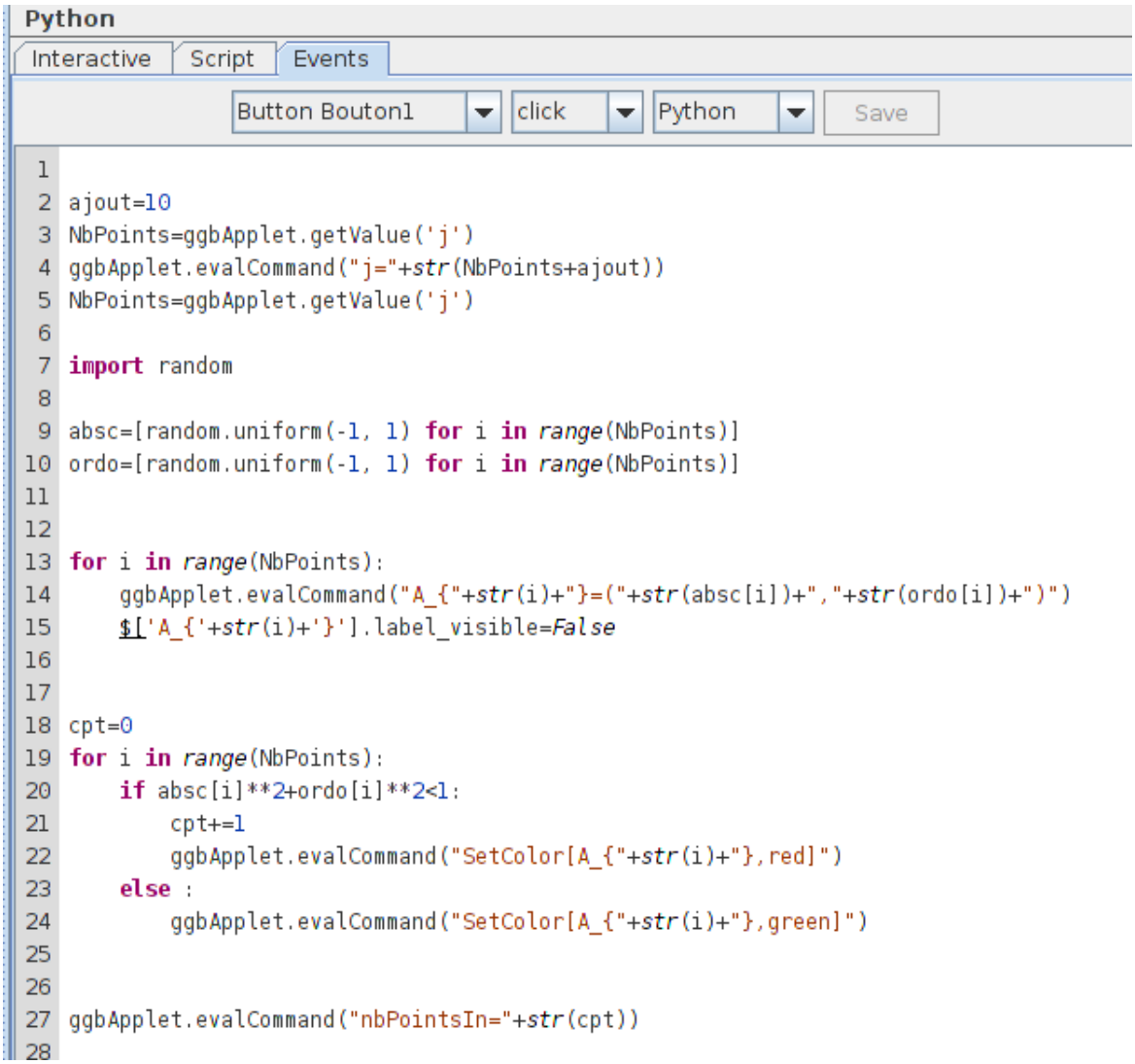


3.1 Variante avec un bouton

Plutôt qu'une modification du nombre de points par un curseur, on peut décider d'attribuer ce rôle à un bouton.
On ajoute donc un bouton :



On renseigne ici uniquement le champ légende en inscrivant par exemple le texte « Ajout de points ».
On associe un script python à ce bouton (le script associé au curseur j de la partie précédente n'est plus utile ici) :



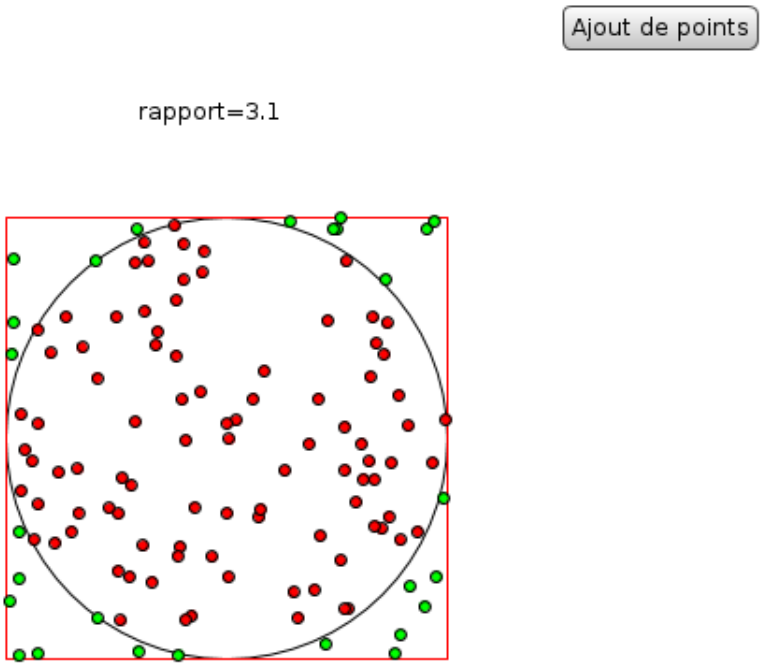

```
1
2  ajout=10
3  NbPoints=ggbApplet.getValue('j')
4  ggbApplet.evalCommand("j="+str(NbPoints+ajout))
5  NbPoints=ggbApplet.getValue('j')
6
7  import random
8
9  absc=[random.uniform(-1, 1) for i in range(NbPoints)]
10 ordo=[random.uniform(-1, 1) for i in range(NbPoints)]
11
12
13 for i in range(NbPoints):
14     ggbApplet.evalCommand("A_{"+str(i)+"}=("+str(absc[i])+","+str(ordo[i])+")")
15     $['A_{'+str(i)+'}'].label_visible=False
16
17
18 cpt=0
19 for i in range(NbPoints):
20     if absc[i]**2+ordo[i]**2<1:
21         cpt+=1
22         ggbApplet.evalCommand("SetColor[A_{"+str(i)+"},red]")
23     else :
24         ggbApplet.evalCommand("SetColor[A_{"+str(i)+"},green]")
25
26
27 ggbApplet.evalCommand("nbPointsIn="+str(cpt))
28
```

Script 6 – PythonGG

```
1  ajout=10
2  NbPoints=ggbApplet.getValue('j')
3  ggbApplet.evalCommand("j="+str(NbPoints+ajout))
4  NbPoints=ggbApplet.getValue('j')
5
6  import random
7
8  absc=[random.uniform(-1, 1) for i in range(NbPoints)]
9  ordo=[random.uniform(-1, 1) for i in range(NbPoints)]
10
11
12 for i in range(NbPoints):
13         ggbApplet.evalCommand("A_{"+str(i)+"}=("+str(absc[i])+","+str(ordo[i])+")")
14         $['A_{'+str(i)+'}'].label_visible=False
15
16
17 cpt=0
18 for i in range(NbPoints):
19         if absc[i]**2+ordo[i]**2<1:
20                 cpt+=1
```

```
                ggbApplet.evalCommand("SetColor[A_{"+str(i)+"},red]")
        else :
                ggbApplet.evalCommand("SetColor[A_{"+str(i)+"},green]")


ggbApplet.evalCommand("nbPointsIn="+str(cpt))
```

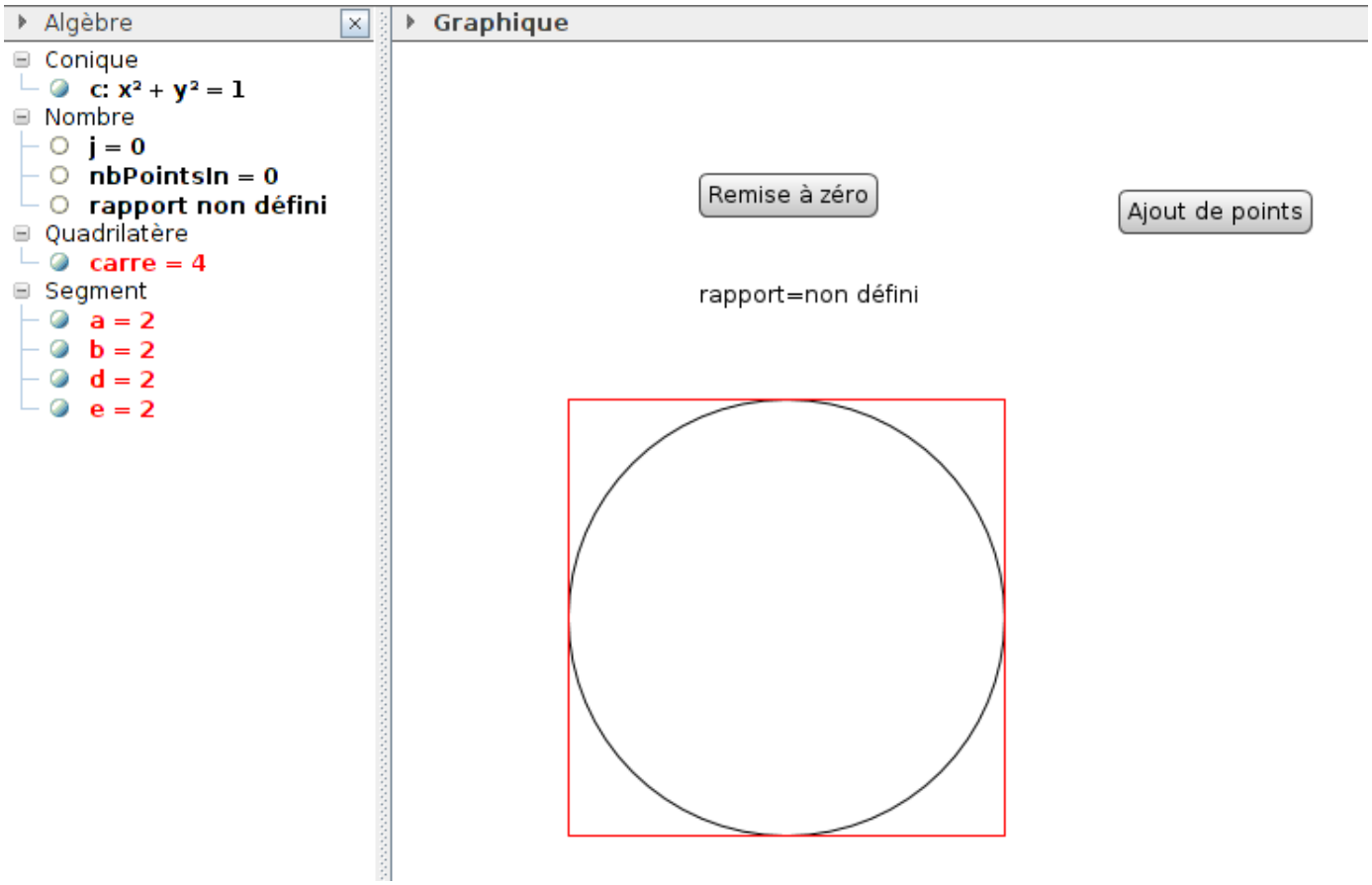
Un click sur le bouton ajoutera alors une série de 'ajout' points.



Il peut dans ce cas être intéressant d'ajouter un bouton de remise à zéro. Pour cette remise à zéro, on pourra associer à ce second bouton le script suivant :

Script 7 – PythonGG

```
NbPoints=ggbApplet.getValue('j')


for i in range(NbPoints):
        if ggbApplet.isDefined("A_{"+str(i)+"}"):
                ggbApplet.evalCommand("Delete(A_{"+str(i)+"})")



ggbApplet.evalCommand("j=0")
ggbApplet.evalCommand("nbPointsIn=0")
```



3.2 Variante avec le bouton

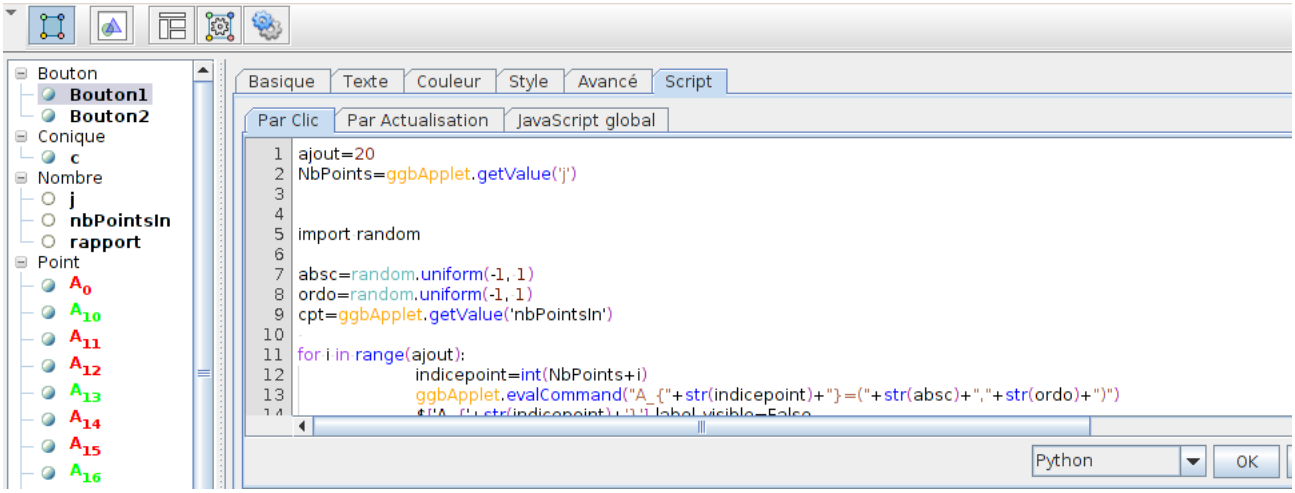
Un défaut de la version précédente est qu'en cliquant sur le bouton, on n'ajoute pas vraiment des points : on recommence le tirage à 0.

Avec la version suivante, on ajoute des points à ceux existants (le bouton de remise à zéro reste inchangé) :

Script 8 – PythonGG

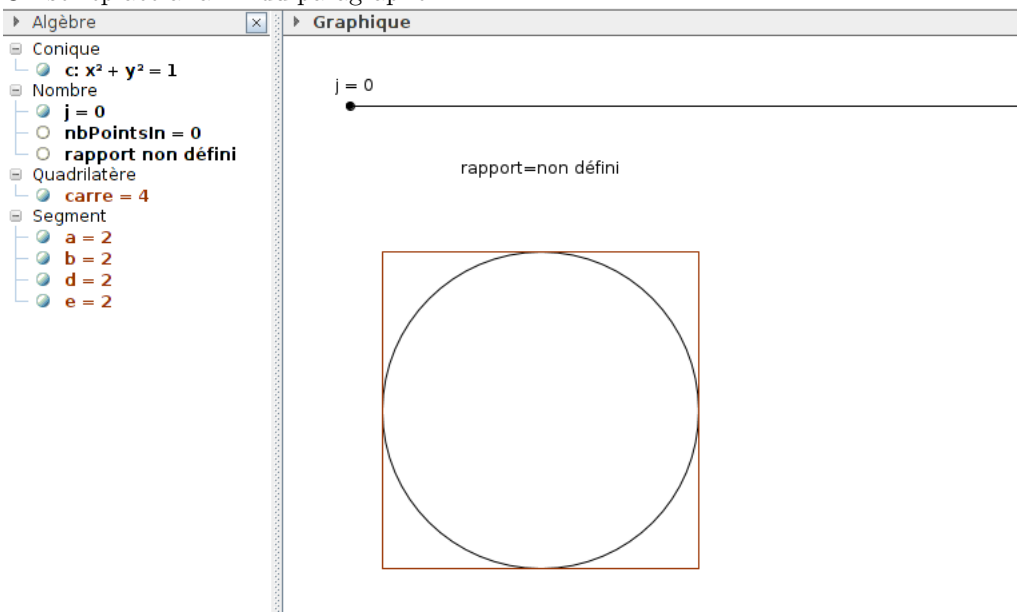
```
ajout=20
NbPoints=ggbApplet.getValue('j')


import random

absc=random.uniform(-1, 1)
ordo=random.uniform(-1, 1)
cpt=ggbApplet.getValue('nbPointsIn')

for i in range(ajout):
        indicepoint=int(NbPoints+i)
        ggbApplet.evalCommand("A_{"+str(indicepoint)+"}=("+str(absc)+","+str(ordo)+")
            ")
        $['A_{'+str(indicepoint)+'}'].label_visible=False
        if absc**2+ordo**2<1:
                cpt+=1
                ggbApplet.evalCommand("SetColor[A_{"+str(indicepoint)+"},red]")
        else :
                ggbApplet.evalCommand("SetColor[A_{"+str(indicepoint)+"},green]")
```

```
        absc=random.uniform(-1, 1)
        ordo=random.uniform(-1, 1)

NbPoints=NbPoints+ajout
ggbApplet.evalCommand("j="+str(NbPoints))
ggbApplet.evalCommand("nbPointsIn="+str(cpt))
```



4 Avec JavaScript

On se replace à la fin du paragraphe 2.



Comme dans le paragraphe 3, nous allons associer un script au curseur j, mais ce script sera cette fois un script javascript.

Pour cela, clic droit sur l'objet j, sélectionner propriétés, aller à l'onglet script et sélectionner javascript.

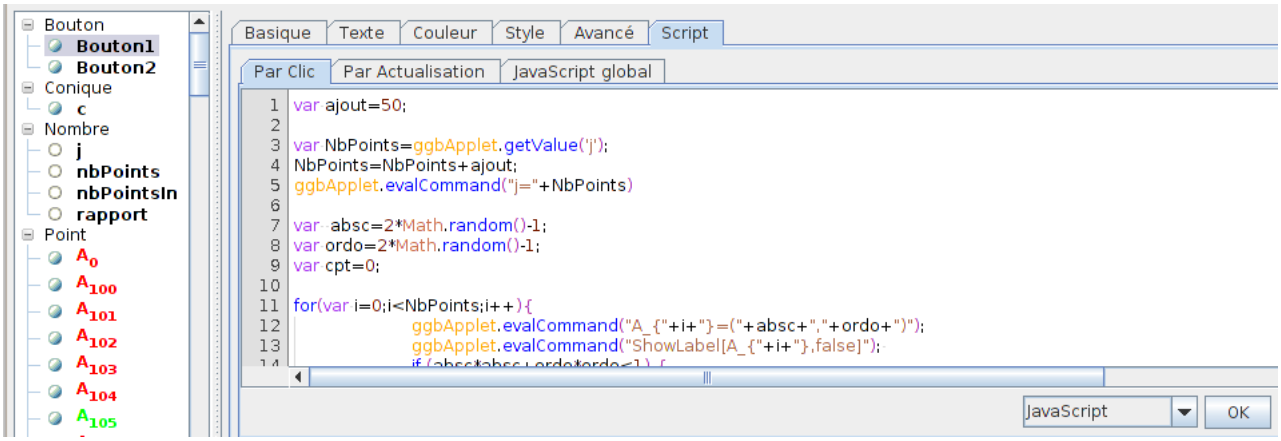
On entre alors le script javascript suivant :

Script 9 – JavaScript

```
NbPoints=ggbApplet.getValue('j');

var  absc=2*Math.random()-1;
var ordo=2*Math.random()-1;
var cpt=0;

for(var i=0;i<NbPoints;i++){
        ggbApplet.evalCommand("A_{"+i+"}=("+absc+","+ordo+")");
        ggbApplet.evalCommand("ShowLabel[A_{"+i+"},false]");
        if (absc*absc+ordo*ordo<1) {
                cpt++;
                ggbApplet.evalCommand("SetColor[A_{"+i+"},red]");
        }
        else ggbApplet.evalCommand("SetColor[A_{"+i+"},green]");
        absc=2*Math.random()-1;
        ordo=2*Math.random()-1;
}
ggbApplet.evalCommand("nbPointsIn="+cpt);

for(var i=NbPoints; i<2000;  i++){
        if (ggbApplet.isDefined("A_{"+i+"}"))  ggbApplet.evalCommand("Delete(A_{"+i
            +"})");
}
```

4.1 Variante avec un bouton

Comme en 3.1, on peut ajouter des points en associant un script javascript à un bouton.
Créons comme en 3.1 deux boutons « Ajout de points » et « Remise à zéro ».
Par un clic droit sur ces boutons, on choisit 'propriétés', onglet 'script', onglet 'par clic' .



On associe au bouton 1 de légende « Ajout de points » le script :

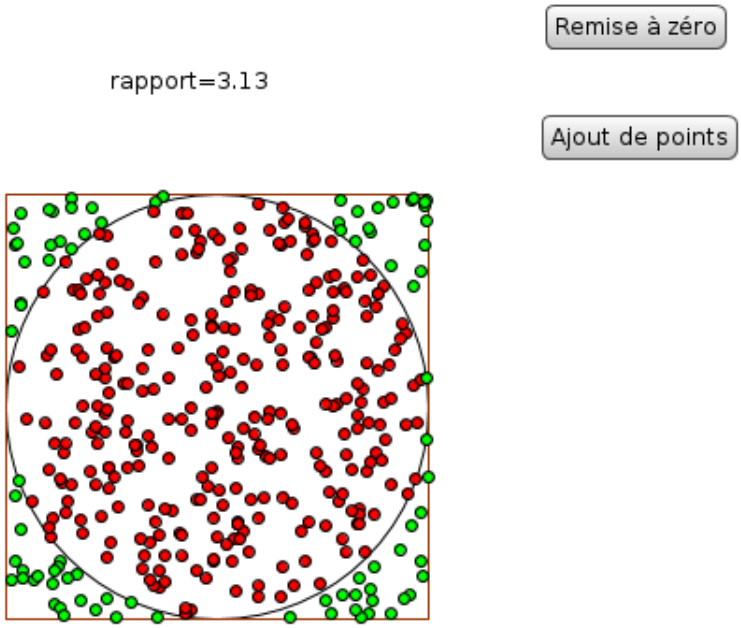
Script 10 – JavaScript

```
var ajout=50;

var NbPoints=ggbApplet.getValue('j');
NbPoints=NbPoints+ajout;
ggbApplet.evalCommand("j="+NbPoints)

var absc=2*Math.random()-1;
var ordo=2*Math.random()-1;
var cpt=0;

for(var i=0;i<NbPoints;i++){
        ggbApplet.evalCommand("A_{"+i+"}=("+absc+","+ordo+")");
        ggbApplet.evalCommand("ShowLabel[A_{"+i+"},false]");
        if (absc*absc+ordo*ordo<1) {
                cpt++;
                ggbApplet.evalCommand("SetColor[A_{"+i+"},red]");
        }
        else ggbApplet.evalCommand("SetColor[A_{"+i+"},green]");
        absc=2*Math.random()-1;
        ordo=2*Math.random()-1;
}
ggbApplet.evalCommand("nbPointsIn="+cpt);
```

On associe au bouton 2 de légende « Remise à zéro » le script :

Script 11 – JavaScript

```
var NbPoints=ggbApplet.getValue('j');

for(var i=0; i<NbPoints; i++){
        if (ggbApplet.isDefined("A_{"+i+"}")) ggbApplet.evalCommand("Delete(A_{"+i
            +"})");
}


ggbApplet.evalCommand("j=0");
ggbApplet.evalCommand("nbPointsIn=0");
```



4.2 Variante sur la version bouton.

La version précédente a le défaut de recréer tous les points à chaque appel plutôt que de réellement ajouter des points.

On peut modifier le script associé au bouton 1 de la façon suivante pour éviter cette création à partir de 0 :

Script 12 – JavaScript

```
var ajout=50;

var NbPoints=ggbApplet.getValue('j');


var  absc=2*Math.random()-1;
var ordo=2*Math.random()-1;
var cpt=ggbApplet.getValue('nbPointsIn');

for(var i=0 ; i<ajout ; i++){
        var indicepoint=i+NbPoints;
        ggbApplet.evalCommand("A_{"+indicepoint+"}=("+absc+","+ordo+")");
        ggbApplet.evalCommand("ShowLabel[A_{"+indicepoint+"},false]");
        if (absc*absc+ordo*ordo<1) {
                cpt++;
                ggbApplet.evalCommand("SetColor[A_{"+indicepoint+"},red]");
        }
        else ggbApplet.evalCommand("SetColor[A_{"+indicepoint+"},green]");
        absc=2*Math.random()-1;
        ordo=2*Math.random()-1;
}
ggbApplet.evalCommand("nbPointsIn="+cpt);

NbPoints=NbPoints+ajout;
ggbApplet.evalCommand("j="+NbPoints);
```

5 Avec des commandes geogebra en ligne de saisie.

Toutes les commandes suivantes sont à saisir dans la ligne de saisie.
On utilise les commandes suivantes (voir wiki.geogebra.org pour les détails de ces commandes) :
— Séquence[<Expression e>, <Variable k>, <de a>, <à b>]
— AléaUniforme[<Min>, <Max>]
— Elément[<Liste>, <Position n>]
— GarderSi[<Condition>, <Liste>]
— Longueur[<Objet>]

1. On commence par définir la variable nbPoints en entrant par exemple

Script 13 – ScriptGG

```
nbPoints=3
```

Cet entier sera le nombre de points tirés au hasard dans notre carré d'aire 4. On pourra l'afficher en tant que curseur et régler de façon appropriée ce curseur.

2. On définit ensuite la liste des abscisses des points :

Script 14 – ScriptGG

```
Labsc=Séquence[AléaUniforme[-1, 1], j, 1, nbPoints]
```

3. On définit de même la liste des ordonnées des points :

Script 15 – ScriptGG

```
Lordo=Séquence[AléaUniforme[-1, 1], j, 1, nbPoints]
```

4. On définit ensuite la liste des points :

Script 16 – ScriptGG

```
Lpoints= Séquence[(Elément[Labsc, j], Elément[Lordo, j]), j, 1, nbPoints]
```

Par un clic droit sur cette liste, on désactive l'affichage de ces points.

5. On définit ensuite la liste des points se trouvant à l'intérieur (strictement ici) du disque :

Script 17 – ScriptGG

```
Lin= GarderSi[x(A)^2 + y(A)^2 < 1, A, Lpoints]
```

Par un clic droit sur cette liste puis propriétés, on affiche en rouge les points de cette liste.

6. On définit ensuite les points qui se trouve à l'extérieur (au sens large ici) :

Script 18 – ScriptGG

```
Lout= GarderSi[x(A)^2 + y(A)^2 >= 1, A, Lpoints]
```

Par un clic droit sur cette liste puis propriétés, on affiche en vert les points de cette liste.

7. On définit ensuite la variable retournant le nombre de points strictement à l'intérieur du disque :

Script 19 – ScriptGG

```
nbPointsIn=Longueur[Lin]
```

8. On définit enfin le rapport :

Script 20 – ScriptGG

```
rapport= 4*nbPointsIn / nbPoints
```

9. Il n'y a plus qu'à modifier la valeur de nbPoints…

6 ScriptGG

6.1 Version 1

Après avoir défini la toile de fond, on peut créer deux boutons.
Un bouton intitulé 'Réinitialisation' qui permettra une remise à 0 et associé au script geogebra suivant (par clic) :

Script 21 – ScriptGG

```
nbPoints=0
Labsc=Séquence[AléaUniforme[-1, 1], j, 1, nbPoints]
Lordo=Séquence[AléaUniforme[-1, 1], j, 1, nbPoints]
Lpoints= Séquence[(Elément[Labsc, j], Elément[Lordo, j]), j, 1, nbPoints]
Lin= GarderSi[x(A)^2 + y(A)^2 < 1, A, Lpoints]
Lout= GarderSi[x(A)^2 + y(A)^2 >= 1, A, Lpoints]
nbPointsIn=0
rapport=0
```

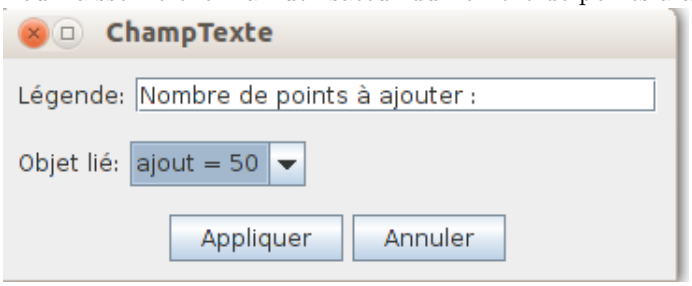
et un bouton 'Ajout de points' associé au script suivant (par clic) :

Script 22 – ScriptGG

```
ajout=50
nbPoints=nbPoints+ajout
Labsc=Séquence[AléaUniforme[-1, 1], j, 1, nbPoints]
Lordo=Séquence[AléaUniforme[-1, 1], j, 1, nbPoints]
Lpoints= Séquence[(Elément[Labsc, j], Elément[Lordo, j]), j, 1, nbPoints]
Lin= GarderSi[x(A)^2 + y(A)^2 < 1, A, Lpoints]
SoitCouleur[Lin,"rouge"]
Lout= GarderSi[x(A)^2 + y(A)^2 >= 1, A, Lpoints]
SoitCouleur[Lout,"vert"]
nbPointsIn=Longueur[Lin]
rapport= 4*nbPointsIn / nbPoints
```

6.2 Version 2

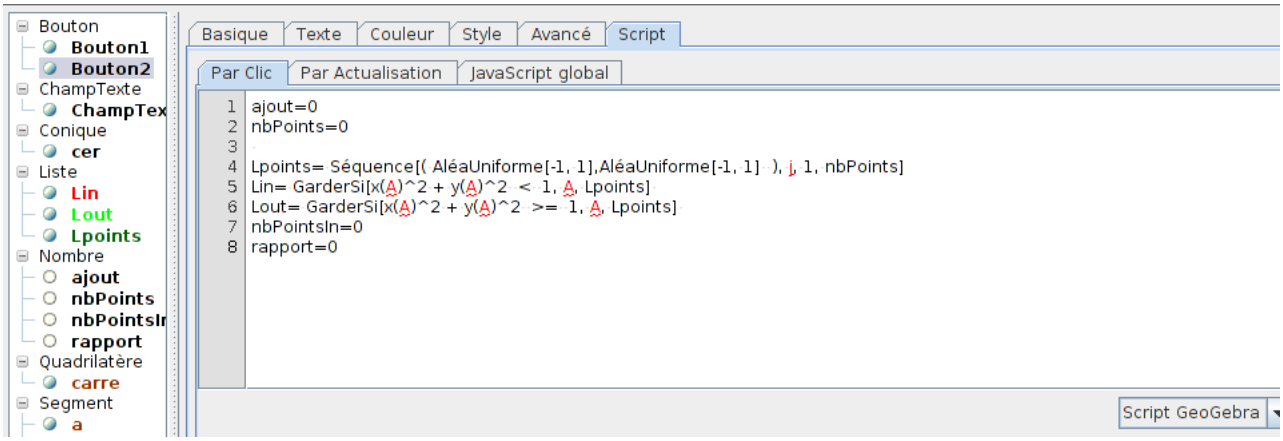
Pour laisser le choix à l'utilisateur du nombre de points à ajouter, on utilise l'outil « insérer un champ de texte »



Le bouton d'intialisation et réinialisation sera associé au script suivant :

Script 23 – ScriptGG

```
ajout=0
nbPoints=0

Lpoints= Séquence[( AléaUniforme[-1, 1],AléaUniforme[-1, 1] ), j, 1, nbPoints]
Lin= GarderSi[x(A)^2 + y(A)^2 < 1, A, Lpoints]
Lout= GarderSi[x(A)^2 + y(A)^2 >= 1, A, Lpoints]
nbPointsIn=0
rapport=0
```



et le bouton « ajout de points » au script suivant :

Script 24 – ScriptGG

```
Lpoints= Union[CopierObjetLibre[Lpoints], Séquence[(AléaUniforme[-1, 1], AléaUniforme
    [-1, 1]), j,  nbPoints+1,  nbPoints+ajout]]
Lin= GarderSi[x(A)^2 + y(A)^2 < 1, A, Lpoints]
SoitCouleur[Lin,"rouge"]
Lout= GarderSi[x(A)^2 + y(A)^2 >= 1, A, Lpoints]
SoitCouleur[Lout,"vert"]

nbPoints=Longueur[Lpoints]
nbPointsIn=Longueur[Lin]
rapport= 4*nbPointsIn / nbPoints
```