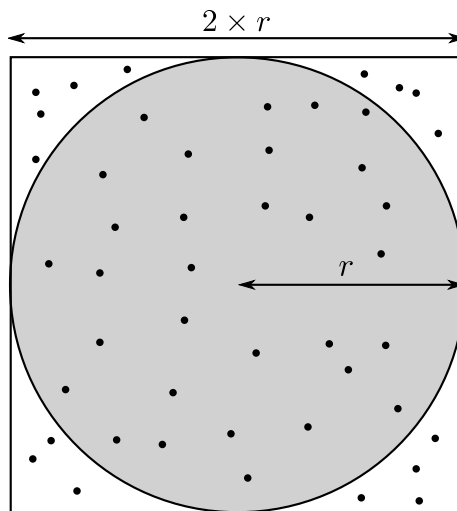


TP1 : CALCUL DE PI VIA UN MONTE CARLO



Dans ce TP, nous considérons une méthode de Monte Carlo afin de calculer π :

- Tracer un cercle de rayon r dans un carré de longueur $2 \times r$
- L'air du cercle est $\pi \times r^2$ et l'air du carré est $4 \times r^2$
- Le ratio des ces deux aires est : $\frac{\pi \times r^2}{4 \times r^2} = \frac{\pi}{4}$
- Si nous générons n points aléatoirement à l'intérieur du carré, alors approximativement $n \times \frac{\pi}{4}$ de ces points (notés m) seront à l'intérieur du cercle
- La valeur π est alors calculée grâce à l'équation :
 - $n \times \frac{\pi}{4} = m$
 - $\frac{\pi}{4} = \frac{m}{n}$
 - $\pi = 4 \times \frac{m}{n}$
- Créer trois fichiers : `main.cc`, `Pi.cc` et `Pi.h`. Le fichier `Pi.h` contiendra l'interface de la classe `Pi` tandis que le fichier `Pi.cc` l'implémentera :

```
//Pi.h
#ifndef PI
#define PI

#include <stdlib.h>

class Pi{
public:
    Pi(int _darts);
    double start();
private:
    const int darts; //Points a simuler
    int score; //Points a l'interieure du cercle
    inline double randZeroToOne(){return rand() / (RAND_MAX + 1.);}
};

#endif
```

```

//Main.cc
#include <iostream>
#include <string>
#include <stdlib.h>
#include "Pi.h"

int main(int argc, char * argv[]){
    if(argc != 2){
        std::cout << "Parameter_1:_number_of_darts_to_calculate_Pi"
        << std::endl;
        return 0;
    }
    std::cout << "Pi_Calculation_-_Sequential_Version:__"
    << argv[1] << "darts" << std::endl;
    Pi pi1(atoi(argv[1]));
    std::cout << "Pi:__" << pi1.start() << std::endl;
    std::cout << "Real_value_of_PI:_3.1415926535897" << std::endl;
}

```

- Implémenter les méthodes du fichier Pi.cc en retournant directement 0 pour la méthode `start()`
- Créer un fichier Makefile contenant les lignes suivantes :

```

COPTIONS = -std=c++11 -O3 -Wall -Wextra -lpthread
LOPTIONS = -Wno-literal-suffix -std=c++11 -Wno-write-strings

main : main.o Pi.o
    g++ $(LOPTION) -o pi main.o Pi.o

main.o : main.cc
    g++ $(COPTION) -c main.cc

Pi.o : Pi.cc Pi.h
    g++ $(COPTION) -Wall -c Pi.cc

clean:
    rm -rf *.o

```

- Quelles sont les significations des variables COPTION et LOPTION (expliquer chaque options)?
- Quelle option faut il mettre pour utiliser les *threads* C++ 2011?
- Les options dans COPTION et LOPTIONS sont t'elles mal placées? Si oui, replacer les biens.
- Compiler ...
- Trouver un algorithme séquentiel pour calculer Pi en effectuant des simulations de points dans l'air du carré
- Implémenter votre algorithme dans la méthode `start()`, elle doit retourner une approximation de Pi
- Tester votre programme en simulant 5000, 50000, 500000, 5000000 et 50000000 points. Que remarquez vous?
- Trouver un algorithme parallèle multi-threads
- Implémenter cet algorithme parallèle pour finir le TP