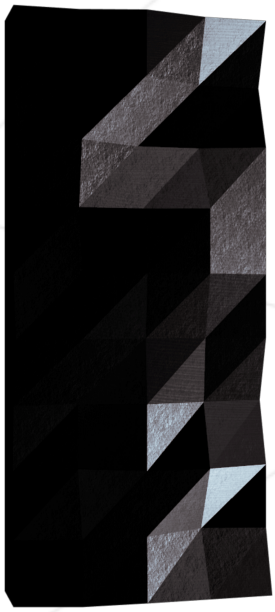

Filler

Pedago pedago@42.fr

Summary: Create your player program to compete against other students on the famous (or not) Filler board. The principle is simple: two players take on each other on a board, and take turns placing the piece that the master of the game (supplied in the form of a Ruby executable) gives them, earning points. The game stops as soon as a piece can no longer be placed. Little playful project!

Contents

Chapter I

Forewords

Many people are asking, “What are the seven deadly sins?” The seven deadly sins viewed by society and literature are:

- Lust (Luxuria in Latin) – to have an intense desire or need: “But I tell you that anyone who looks at a woman lustfully has already committed adultery with her in his heart” (Matthew 5:28).
- Gluttony (Gula in Latin) – excess in eating and drinking: “for drunkards and gluttons become poor, and drowsiness clothes them in rags” (Proverbs 23:21).
- Greed (avaritia in Latin)- excessive or reprehensible acquisitiveness: “Having lost all sensitivity, they have given themselves over to sensuality so as to indulge in every kind of impurity, with a continual lust for more” (Ephesians 4:19).
- Laziness (Acedia in Latin) – disinclined to activity or exertion: not energetic or vigorous: “The way of the sluggard is blocked with thorns, but the path of the upright is a highway” But it is mostly about the Laziness of the soul, boredom, to walk away from God, prayers. (Proverbs 15:19).
- Wrath (Ira in Latin) – Also known as Anger. Strong vengeful anger or indignation: “A gentle answer turns away wrath, but a harsh word stirs up anger” (Proverbs 15:1).
- Envy (Invidia in Latin) – painful or resentful awareness of an advantage enjoyed by another joined with a desire to possess the same advantage: “Therefore, rid yourselves of all malice and all deceit, hypocrisy, envy, and slander of every kind. Like newborn babies, crave pure spiritual milk, so that by it you may grow up in your salvation” (1 Peter 2:1-2).
- Pride (superbia in Latin)- quality or state of being proud – inordinate self esteem: “Pride goes before destruction, a haughty spirit before a fall” (Proverbs 16:18).

Chapter II

Introduction

Filler is an algorithmic game which consists in filling a grid of a known size in advance with pieces of random size and shapes, without the pieces being stacked more than one square above each other and without them exceeding the grid. If one of these conditions is not met, the game stops.

Each successfully placed piece yields a number of points, and has only one player, the goal of the game could be to get the best score possible. However, it is with two players that the filler takes all his interest. Each player has for the purpose of placing as many pieces as possible while attempting to prevent his opponent from doing the same. At the end of the game, the one with the most points wins the match...

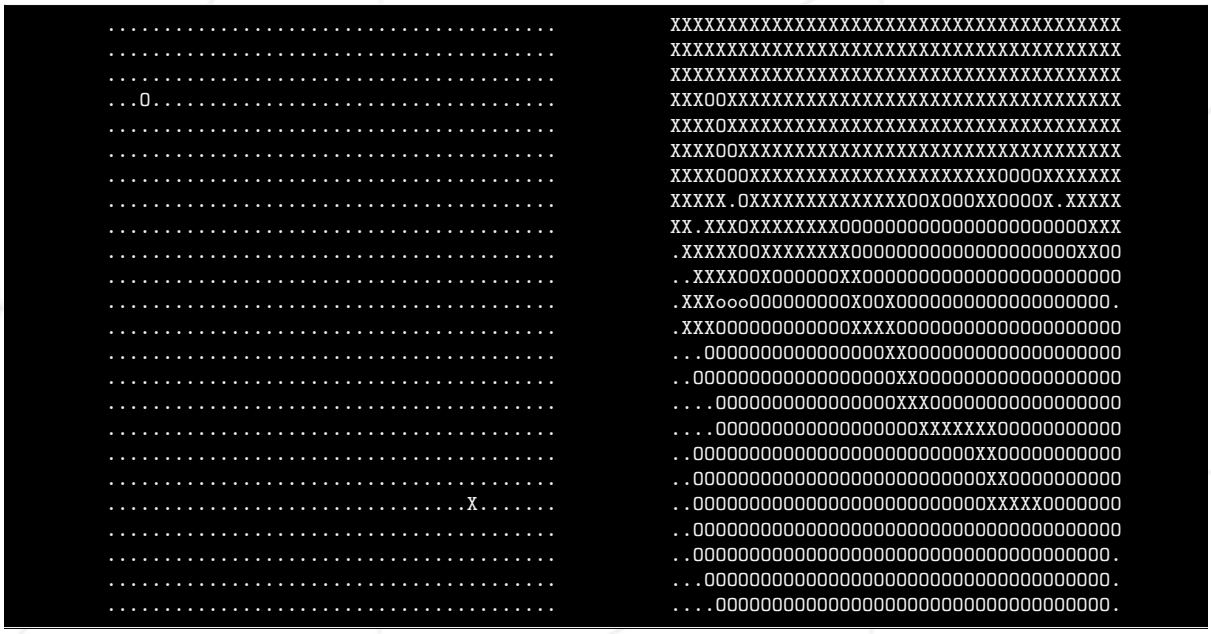

Figure II.1: On the left: the initial state of the grid, at the right: the result after the fight between two players.

Chapter III

Goals

Writing a Filler player is a very interesting algorithmic challenge. Each turn, the active player receives the grid status and must maximize his points while trying to minimize those of the opponent by eliminating it as quickly as possible.

The objectives of this project always bring together the usual objectives of early curriculum projects: rigour, practice of C and practice of elementary algorithms. But unlike a simpler game like Fillit, it's no longer just a matter of arranging your pieces as efficiently as possible, but now of preventing your opponent from doing so! You will therefore have to create your own filling algorithm to counter the enemy algorithm.

You must master the VM provided with this subject too...

Chapter IV

General Instructions

- This project will only be corrected by humans. You are therefore free to organize and name your files as you wish, while respecting the constraints listed here.
- You'll have to submit a file called `author` containing your username followed by a '\n' at the root of your repository.

```
$>cat -e author
xlogin$
```

- The executable file must be named `<login>.filler`.
- It must be at the root of the repository.
- You must submit a `Makefile`.
- Your `Makefile` must compile the project and must contain the usual rules. It must recompile and re-link the program only if necessary.
- If you are clever, you will use your library for your player. Submit also your folder `libft` including its own `Makefile` at the root of your repository. Your `Makefile` will have to compile the library, and then compile your project.
- Your project must be written in accordance with the Norm.
- You have to handle errors carefully. In no way can your program quit in an unexpected manner (Segmentation fault, bus error, double free, etc).
- Within your mandatory part, you are allowed to use the following functions:
  - `read`
  - `write`
  - `malloc`
  - `free`
  - `perror`

    - `strerror`

- Good luck and GOOD FIGHT to all!

Chapter V

Mandatory part

V.1 The Filler

- In this game, two players fight each other. They play one after the other.
- The goal is to collect as many points as possible by placing the highest number of pieces on the the game board.
- The board is defined by X columns and N lines, it will then become X*N cells.
- At the beginning of each turn, you will receive your game piece.
- A game piece is defined by X columns and N lines, so it will be X*N cells. Inside each game piece, will be included a shape of one or many cells.
- To be able to place a piece on the board, it is mandatory that one, and only one cell of the shape (in your piece) covers the cell of a shape placed previously (your territory).
- The shape must not exceed the dimensions of the board
- When the game starts, the board already contains one shape.
- The game stops at the first error: either when a game piece cannot be placed anymore or it has been wrongly placed.

V.2 The Board

A board is a two-dimensional grid with an arbitrary number of rows and columns. To launch the game an initial board must be passed as an argument to the VM. This initial board must have a starting form for each player.

Here is an example of an initial board of 14 by 30 :

```
Plateau 14 30:
    012345678901234567890123456789
000 ..............................
001 ..............................
002 ..X...........................
003 ..............................
004 ..............................
005 ..............................
006 ..............................
007 ..............................
008 ..............................
009 ..............................
010 ..............................
011 ............................O..
012 ..............................
013 ..............................
```

V.3 The tokens

The tokens are managed randomly by the VM. You can't predict their size or shape until the VM transmits them to your program. Here are some arbitrary examples of possible tokens to give you an idea:

```
Piece 4 7   Piece 4 5:  Piece 3 6:
 ...*...      .**..       .****.
 ...*...      .***.       **....
 ...*...      ..*..       *.....
 ..***..      .....
```

V.4 The Topic

V.4.1 The Player

- The executable that will enable you to play the filler is attached to this subject.
- For this project, you will have to create a filler player. Your goal is to win:
  - It will read the board and the game pieces placed on the standard output.
  - Each turn the filler rewrites the board map and includes a new piece to be placed.
  - In order to place the game piece on the board, the player will have to write it's coordinates on the standard ouput.
  - The following format must be used "X Y\n".

- You will collect points each time you place a piece.

```
Plateau 14 30:
012345678901234567890123456789
000 ..............................
001 ..............................
002 ..............................
003 ..............................
004 ..............................
005 ..............................
006 ..............................
007 ..............................
008 ..............O...............
009 ..............................
010 ..............................
011 ..............................
012 ..............................
013 ..............................
Piece 4 7:
...*...
...*...
...*...
..***..
```

Watch out! You must write the coordinates of the token and not those of the shape.

V.4.2 Multi Players

- Player number:
  - The first two lines of the filler must be in the following format:

```
$$$ exec pPLAYER_NUMBER : [PLAYER_NAME]
```

  - The filler will only send the line that concerns your program. You'll have to get your player number.
  - If you are Player 1 your program will be represented by "o" and "O". If you are Player 2, your program will be represented by "x" and "X". The first step will be to get your player number.
  - The lowercases ("x" or "o") will highlight the piece last placed on the board. At the following turns, that same piece will be represented by the uppercase letters ("X" or "O"), as it won't be the piece last placed anymore.
  - You will collect points each time you place a piece.
- How the game works
  - At each turn, the filler will send the updated map and a new token to the player concerned.
  - The player concerned will write on the standard output his or her piece's coordinates to place it on the board.

- The filler will send the map and a new piece to the other player.

V.4.3 How the game rolls

- Here is an example on how a game will roll out.

```
$>./filler_vm -p1 user1 -p2 user2 -v -f samples/w1.flr
$$$ exec p1 : [user1]
$$$ exec p2 : [user2]
Plateau 14 30:
01234567890123456789012345678 9
000 ..............................
001 ..............................
002 ..............................
003 ..............................
004 ......X.......................
005 ..............................
006 ..............................
007 .........................O...
008 ..............................
009 ..............................
010 ..............................
011 ..............................
012 ..............................
013 ..............................
Piece 3 6:
.****.
**....
*.....
<got (O) : [7 24] (7,24)
Plateau 14 30:
01234567890123456789012345678 9
000 ..............................
001 ..............................
002 ..............................
003 ..............................
004 ......X.......................
005 ..............................
006 ..............................
007 .........................oooo.
008 .......................oo....
009 .......................o......
010 ..............................
011 ..............................
012 ..............................
013 ..............................
Piece 3 8:
......*.
......**
.......*
<got (X) : [4 0] (4,0)
Plateau 14 30:
01234567890123456789012345678 9
000 ..............................
001 ..............................
002 ..............................
003 ..............................
004 ......x.......................
005 ......xx......................
006 .......x......................
007 .........................OOOO.
008 .......................OO....
009 .......................O......
010 ..............................
011 ..............................
012 ..............................
013 ..............................
[...]
== X fin : 175 [1018918090]
== O fin : 168 [1018918090]
```

V.4.4 VM

If you experience problems with the VM, please contact us on slack. Really make sure the problem is indeed coming from the VM and not from your program.

Chapter VI

Bonus part

Bonuses will only be evaluated if your mandatory game is PERFECT. By PERFECT, we mean of course that it is fully realized, and that it is not possible to put its behavior in default, even in case of error as vicious as it is, misuse, etc. Concretely this means that if your mandatory game does not get ALL points the rating, your bonuses will be fully IGNORED.

As bonus points, we will take into account:

- A graphic visualizer.
- Any additional bonuses that you will consider useful and that your peers will approve and enjoy.

Chapter VII

Submission and peer correction

Submit your work on your `GiT` repository as usual. Only the work on your repository will be graded.