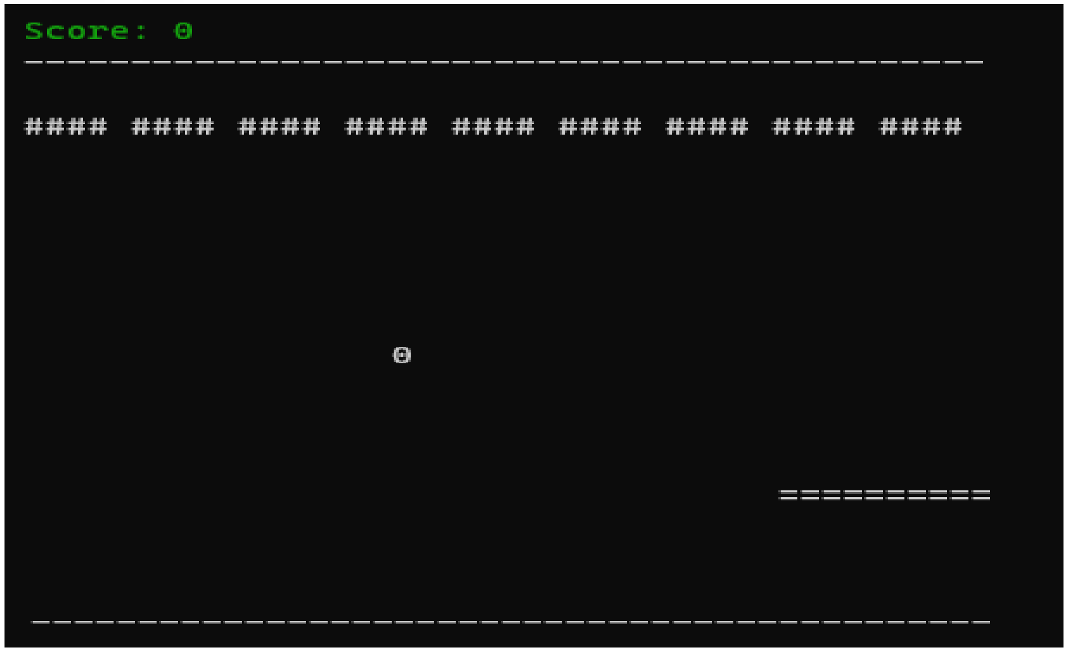
FINAL PROJET – BICKS BREAKING GAME

PROGRAMMING HASKELL 4.0

Titre de la présentation

PROGRAMMING HASKELL 4.0



```
Score: 0
---------------------------------------------

#### #### #### #### #### #### #### #### ####




                 o




                                   ==========



---------------------------------------------
```

SUMMARY

- Background & objective
- Project description
- Description of the Solution
- Solution and Approach - Playing

Background & objective

Write efficient algorithms to create a gaming program.

Understand and use various constructs of the programming language of such as :

- Conditionals,
- Recursion ,
- Currying,
- Monads

Assignement

Availability

- Please make the project publicly available as a version-controlled repository in GitHub, GitLab, SourceHut, etc. Mailed submissions of tar files or zip files ARE NOT acceptable.

- The project must be a `cabal` or `stack` project rather than just a single file or a collection of Haskell files. Please refer to the relevant sessions

  and `https://cabal.readthedocs.io/` documentation for setting up the project.

Complexity

- The project should be moderately extensive. Try to hit a little above Tic Hask Toe example. If you're uncertain about the complexity, just consult me before you start working on the project.

- The project should demonstrate the concepts which we learnt in the module. You can't avoid using IO because you'd need to do IO. Try to use features like higher-order functions, monads, and transformers both for convenience and learnability.

Deadline

- There is NO hard deadline for the submission of the project but I'd encourage you to present the project on 8 October 2023. If your project is not complete before the presentation deadline, you can also present an unfinished but presentable project and submit later.

Assignement

Deadline

- There is NO hard deadline for the submission of the project but I'd encourage you to present the project on 8 October 2023. If your project is not complete before the presentation deadline, you can also present an unfinished but presentable project and submit later.

- Although I'd encourage you to complete the project before the presentation, if you're unable to present, you can also submit anytime later. You can also record a screencast and post it in the Telegram group for us to see.

- The project would be presented in the class with a demonstration of the working program and a high-level walk-through of the code. You may have to answer questions from fellow learners about the project design and implementation after the presentation. Don't worry, it's not supposed to be a grilling session, but just a casual discussion where we might discuss and add some comments.

Assignement

Tips

- Try to focus on the types in your programs. When you're new to Haskell, working through the type errors while compiling the code can be extremely frustrating. But don't be discouraged by it. When you get a type error, before reading and trying to understand the error, just go back and look into the code. I know, it sounds wrong but it works! Maybe you missed a comma, or maybe you forgot to add a parameter in the type declaration. With time and practice, you'd get an intuition for many confusing error messages rarely point to the exact error. But when your code finally compiles, it'd be rewarding.

- Try to build incrementally. If you're building a simple game, start with a lesser variant which is simpler to implement. Let's say you're trying to implement Tetris. First try to just implement the arena and figure out how one single block can be animated. Once you've got the principles of the implementation down, adding new features would be very easy.

- It's very important to isolate the logic and the effects in your program by proper use of IO and State, eg, in the Tetris game, the function to render the Tetris arena on the screen should be decoupled from the function to check if the game is over or the function to remove the filled lines. Functions which deal with state should not deal with IO and vice versa.

Design of the Solution

Global Views :5 Files for the game source codes

1. GameConfig – hold the configuration
2. GameTypes – hold the data types
3. GameRenderCLI – To render the screen
4. GameInput – To get Input of the play
5. Main – File to run the game
6. Bricksbreak.cabal

Description of the solution

- ✓ Initialization
- ✓ Bricks,Lives, Ball creation
- ✓ Collision Ball & Racket
- ✓ Collision Ball & Bricks
- ✓ Collision Windows borders
- ✓ Bricks breaking
- ✓ Game update
- ✓ Lives , Bricks and Ball displays
- ✓ Game Ending

Description of the game : How It works ?

The brick-breaking game consists of destroying bricks. For this, the player is equipped with a racket allowing him to hit a ball.

When he misses the ball, he loses the party. When he breaks all the bricks He wins the game with his score displayed on the top left

N.B:
Set the letter in the "GameInputs file:

- Line 22 to move the cursor to the left
- Line 25 to move the cursor to the right

When Playing, after pressing the letter hit enter to move the cursor