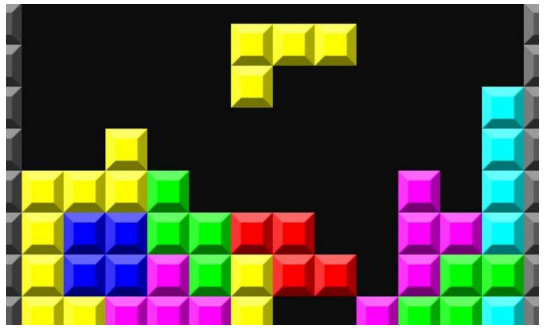
Compiler Construction Lab:
Game Language Design and Implementation

Prof. Kunal Korgaonkar, Prof. Soumyadip Bandyopadhyay
and Prof. Ramprasad Joshi

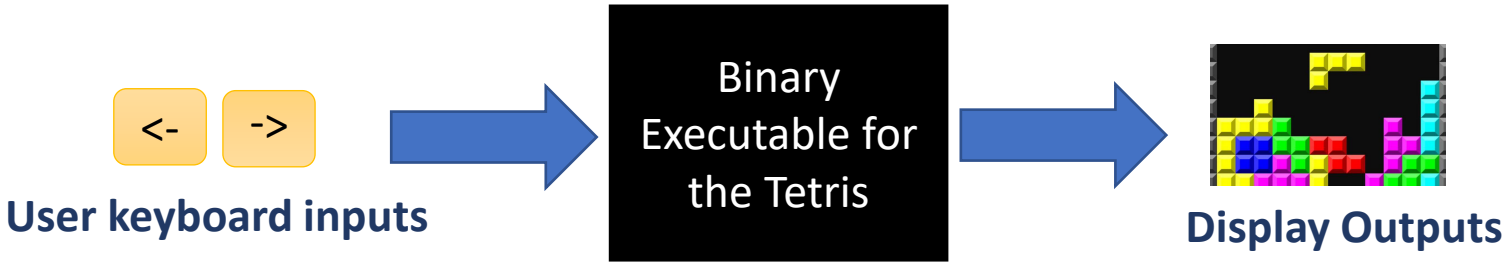
Computer Science and Information Systems
Bits Pilani Goa Campus, India

Objective of this course

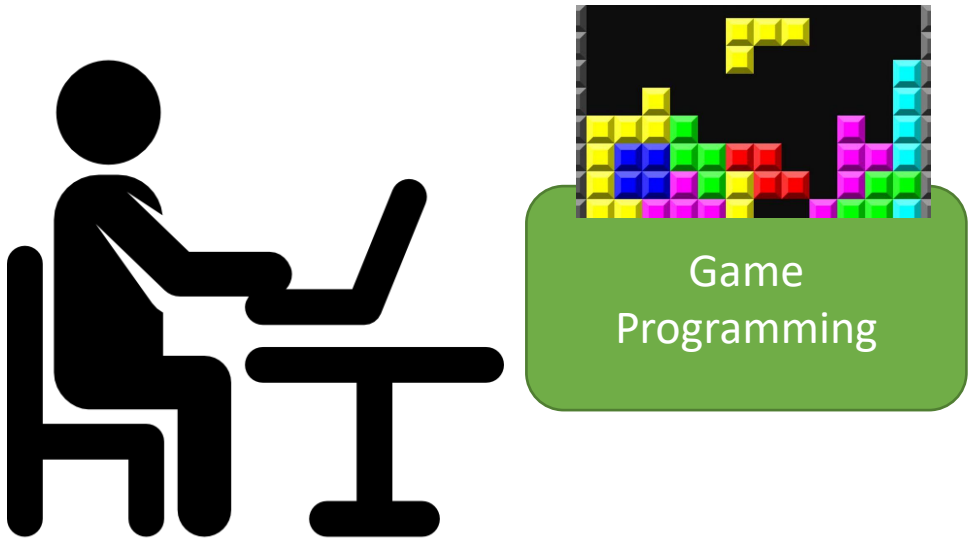
- First, design your own game programming language for the Tetris game
- Second, compile a program written in your game programming language to final executable form
- Steps for compiling,
  Step 1: Game-Tetris-Program-X ➔
  Python (or C++ or C Program) as Language for Intermediate Code
  Step 2: Python (or C++ or C Program) as Intermediate Code ➔
  Binary Executable for Tetris (trivial)



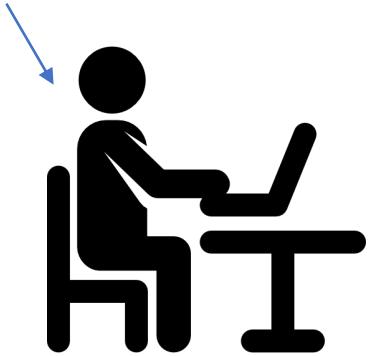
Tetris



Programming the Game vs Programming the Game Programming Language



This is also you, but later!

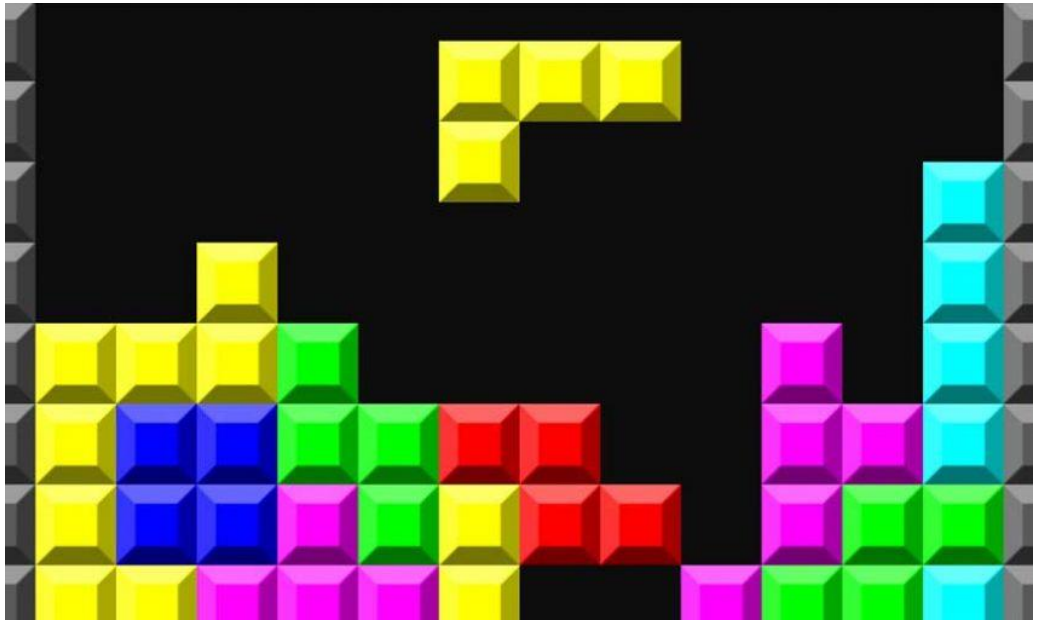


This is you!

↳ This is the task.

Game
Programming
Compiler

2D World Design to be Created



- The end programmer should be able to create a 2D world
- The 2D world should be realistic and active
  → our task is to give the programmer features to make this.
- Many more variants of the game should be possible and accommodated in the game programming language

Some Requirements of the Game

- The language defines and interprets user inputs (L and R arrow key)
- The language defines and produce the output 2D world
- The language defines a sense of movement of blocks in time
- The language supports any shape blocks
- The language supports randomness in the shape of the blocks
- The language supports rules such that laws of physics are not violated
- What else .. ?

Why Choose this Problem for the Course?

Game programming design is interesting! Why?

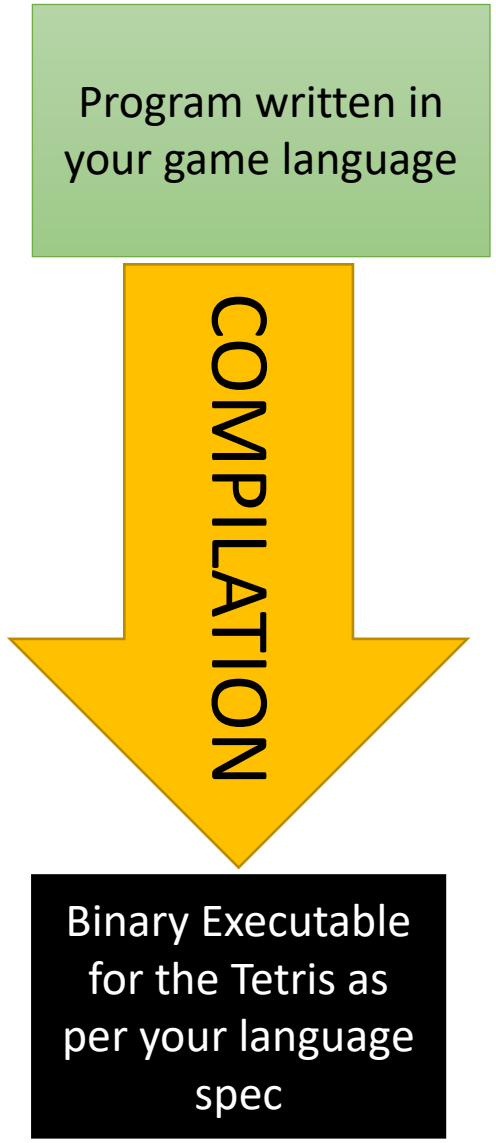
Language related questions

- Interesting questions on what should go in the language and what should not?
- Compile statically or dynamically?
- Interpreted style language or not ?
- What to make the part of the language and what is to be made as some inputs to the program (e.g. speed of blocks)

Language interpretation and code generation questions

- How would one, break down the problem of taking a program in the language and generating the final executable?
- What would be the stages of compilation or steps during and interpretation?

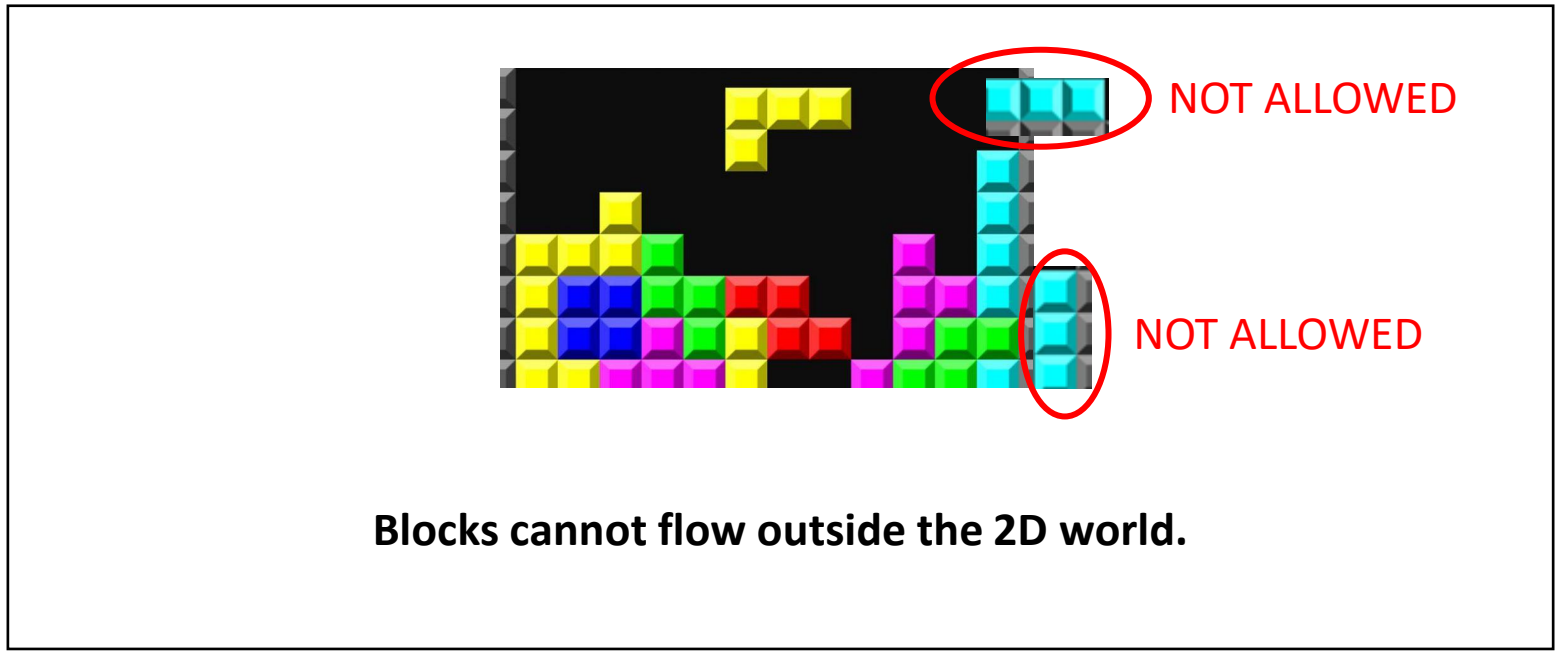
Possible Stages



1. Scan the program
2. Parse the program
3. Translation to intermediate code output
4. Intermediate code to final executable should be trivial (trivial)

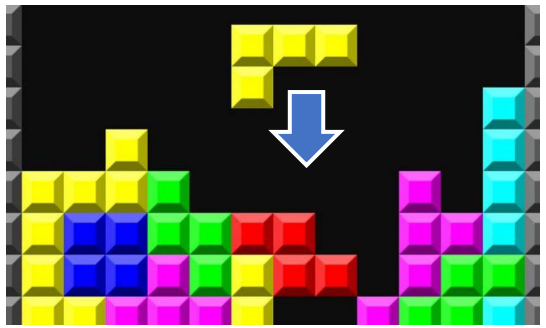
Idea(s) about Step 2: Parse the program

- For instance, parser can check if the program is generating valid states of the 2D world



Idea(s) about Step 3: Translation to intermediate code

- For instance the translator can internally generate the next state of the 2D world given the current program state and current user input



How to update the 2D world and when.

Conclusion

- Programming language design and implementation problem
- Choose the language wisely since that step is early in the design flow; design flaws are hard to rectify later
- Choose either interpreted or non-interpreted language (your choice)
- Be flexible in the language design so that many programmer requirements are accommodated
- Scanning, parsing and translation are important steps
- *Enjoy the programming assignments! In a few years time you might be designing a new domain specific language.*