GRIDLang

Grid Based Game Programming Language

PLT Spring 2017

Team

```
Player Akshay_Nagpal,
       Dhruv_Shekhawat,
       Parth_Panchmatia,
       Sagar_Damani ;
```

Main Goals

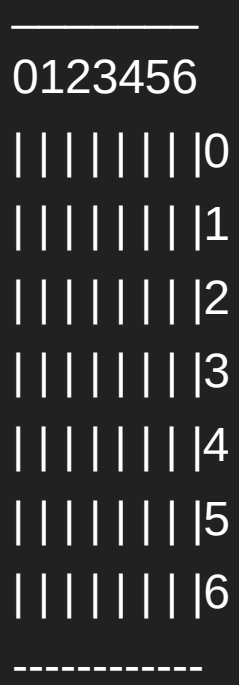
- Design games in an intuitive and expressive manner
- Quickly prototype grid-based games and get a programmatic view
- Simplify the process of :
  - defining rules for a game
  - grid creation and manipulation
  - in-built language components that enable programmer to express more with less lines of code

Core Features

- Strongly typed
- Move Driven
- Structs, Pointers, Arrays(1D & 2D)
- Standard Library

Initialize Grid

```
Grid_Init<7,7>;
```



Creating Player and item structs

```
Player
{
    Piece horse h1,h2,h3;
    int score;
}
Piece horse
{
    int value;
}
```

Adding to Grid

```
Player p1;

int setup(){
        p1.h1.displayString = "h1";
        p1.h2.displayString = "h2";
        p1.h3.displayString = "h3";
        Grid<3,6> <-- p1.h1;
        Grid<3,2> <-- p1.h2;
        Grid<5,2> <-- p1.h3;
        return 0;

}

printGrid();
```

```
_________________________
0__1__2__3__4__5__6_
|  |  |  |  |  |  |  |0
|  |  |  |  |  |  |  |1
|  |  |  |  |  |  |  |2
|  |  |h2|  |  |  |h1|3
|  |  |  |  |  |  |  |4
|  |  |h3|  |  |  |  |5
|  |  |  |  |  |  |  |6
---------------------
```

Grid Initialization

NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL

GenericPiece from MiniChess

```
Piece GenericPiece
{
    Piece King* King_node;
    Piece Pawn* Pawn_node;
    Piece Bishop* Bishop_node;
    int x, y ;
    Piece GenericPiece* next ;
    string nametag, typetag ;
    Player* owner ;
}
```

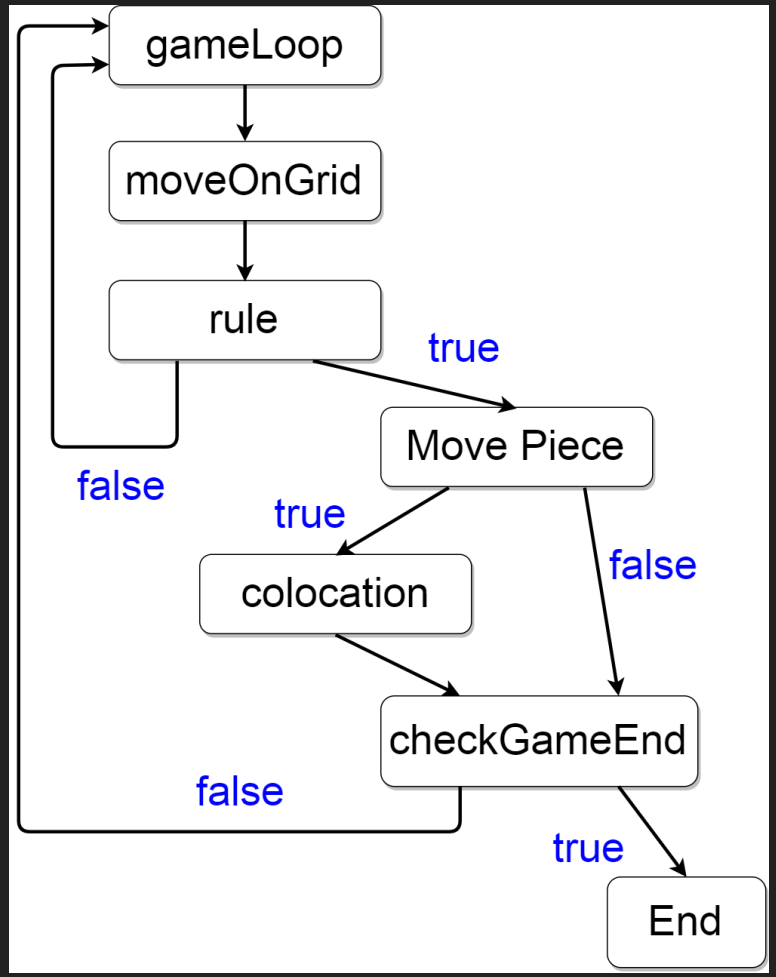
```
Piece King
{
    // programmer's code
}

Piece Pawn
{
   // programmer's code
}

Piece Bishop
{
   // programmer's code
}
```

NULL	NULL	NULL
NULL	**Piece* horse h_node;** **Piece* bishop b_node = b1;** **typetag = “bishop”** **nametag = “b1”** **owner = “black”**	NULL
NULL	NULL	NULL

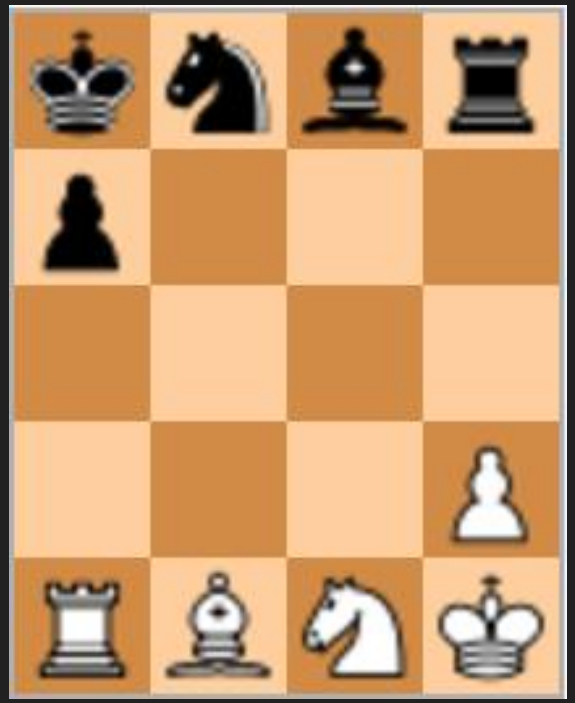
NULL	NULL	NULL
NULL	Piece* horse h_node; Piece* bishop b_node = b1; typetag = “bishop” nametag = “b1” owner = “black”	NULL **next**
NULL	NULL	NULL

Piece* horse h_node = h1;
Piece* bishop b_node;
typetag = “horse”
nametag = “h1”
owner = “black”

Control Flow

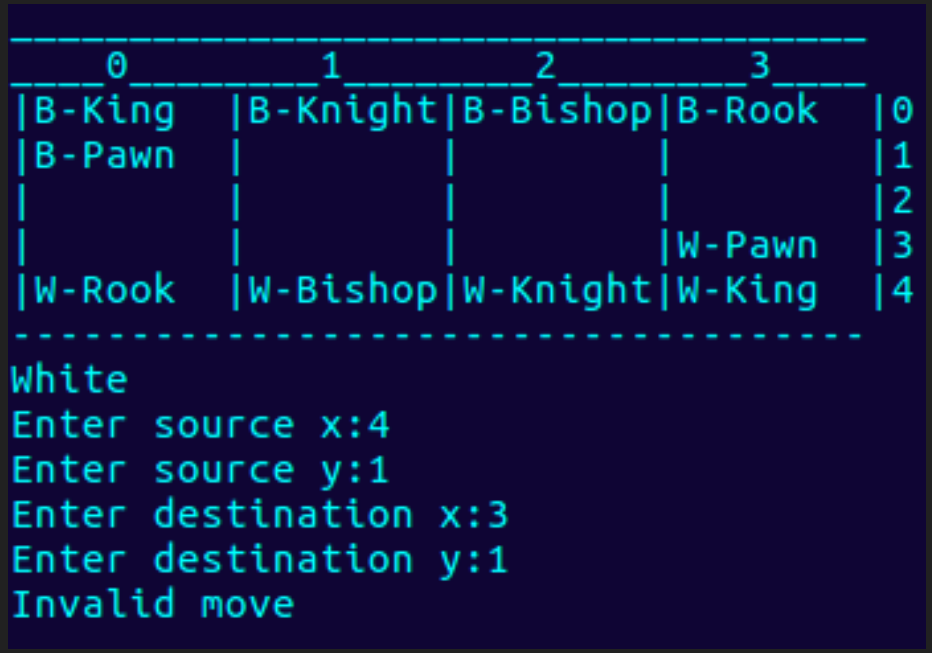


Mini Chess



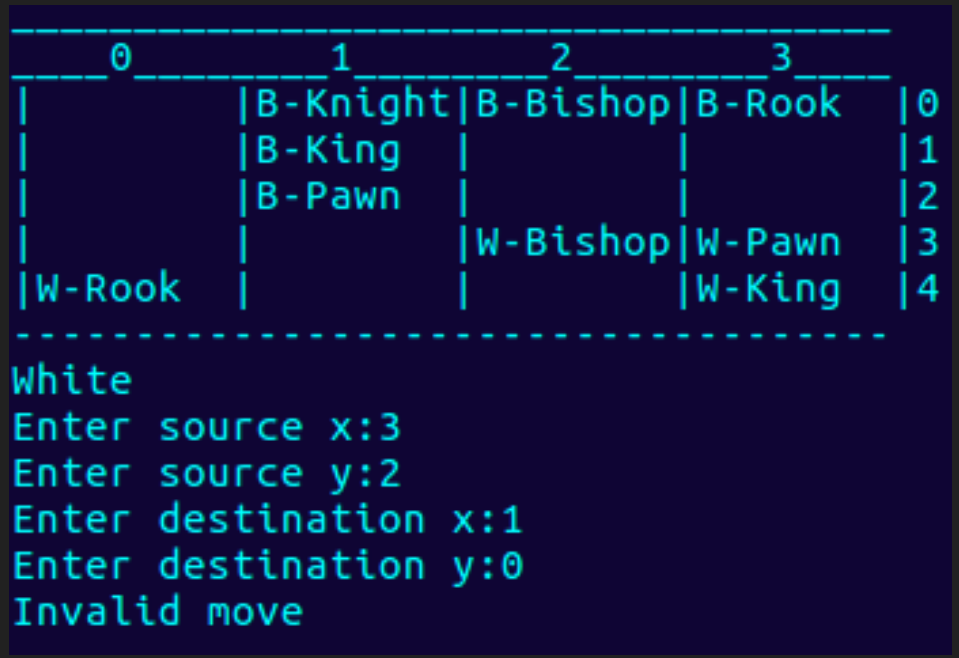
Bishop Rule - Check if Move is Diagonal

```
if(abs(dst_x - src_x) == abs(dst_y - src_y))
```



Bishop Rule – Check if Diagonal is Blocked

```
if (traverse(src_x, src_y, dst_x, dst_y) == 1) {

        return 0;

}
```

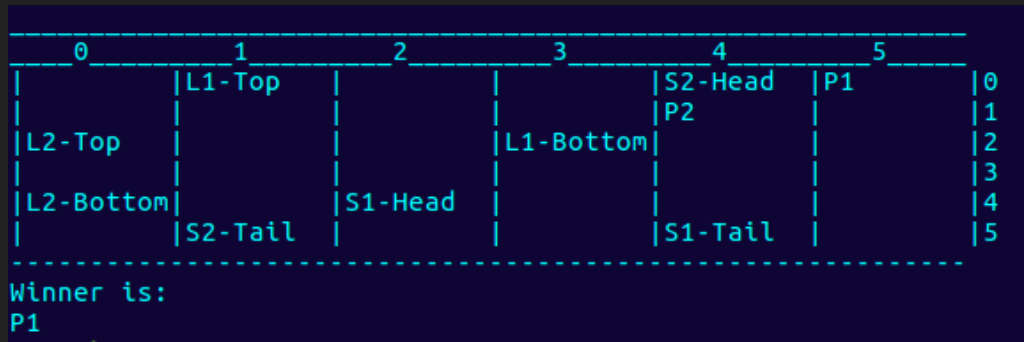


Colocation

```
int colocation(int x, int y, Piece GenericPiece* i1, Piece GenericPiece* i2)
{
    deleteFromGrid(x,y,i2.nametag);
    return 0;
}
```

checkGameEnd (Snakes and Ladders)

```
int checkGameEnd()
{
    Piece Token *t;
    Piece GenericPiece *token;
    t = getCurrentPlayer();
    token = getPieceFromGrid(t.displayString);
    if (token.x == 0 && token.y == 5){
        printGrid();
        print("Winner is: ");
        print(t.displayString);
        return 1;
    }
    return 0;
}
```



Lessons Learned

- Have a concrete plan of what your language does.
- Team matters a lot. Choose team members based on their ability to learn.
- Two heads are better than one.