*Character classes*

```
decimal              -> "0" | "1" | ... | "9"
lowercase            -> "a" | "b" | ... | "z"
uppercase            -> "A" | "B" | ... | "Z"
anycase              -> lowercase | uppercase
unichar              -> any unicode character
strchar              -> unichar except for "\\" and "\""
```

*Reserved*

```
keyword              -> "game" | "piece" | "this" | "width" | "height"
                      | "title" | "players" | "turnOrder" | "board"
                      | "grid" | "setup" | "wall" | "name" | "possibleDrops"
                      | "possibleMoves" | "winCondition" | "tieCondition"
operator             -> "and" | "or"
pattern_keyword      -> "friend" | "foe" | "this" | "empty"
pattern_operator     -> "*" | "?" | "+" | "!"
```

*Literals*

```
integer              -> decimal{decimal}
direction            -> "n" | "s" | "e" | "w" | "ne" | "nw"
                      | "se" | "sw"
coordinate           -> uppercase{uppercase}decimal{decimal}
string               -> "\"" {strchar | "\\" unichar} "\""
```

*Identifiers*

```
function             -> lowercase anycase{anycase}
identifier           -> uppercase {anycase}
variable             -> "$" anycase {anycase}
```

*Program structure*

```
program              -> {function_def} game_decl
function_def         -> "define" function "[" {variable} "]" expression
game_decl            -> "game" declaration_struct
declaration_struct   -> "{" declaration {declaration} "}"
declaration          -> (keyword | identifier) structure
structure            -> declaration_struct | expression
```

*Expressions*

```
expression           -> function_call
                      | element operator expression
                      | if_expr
                      | lambda_expr
                      | element
element                    -> "(" expression ")
                      | variable
                      | list
                      | pattern
```

```
                     | keyword
                     | direction
                     | coordinate
                     | integer
                     | string
                     | identifier
function_call        -> function list
if_expr              -> “if” expression “then” expression “else” expression
lambda_expr          -> “#” {varlist} “=>” expression
list                 -> “[“ {element} “]”
```

*Patterns*

```
pattern              -> “/” pattern_expr “/”
pattern_expr         -> pattern_expr {pattern_expr}*
                     | ( pattern_expr )
                     | ( pattern_expr ) integer
                     | pattern_expr "*"
                     | pattern_expr "?"
                     | pattern_expr "+"
                     | pattern_val
pattern_val          -> direction
                     | pattern_check
                     | “!” pattern_check
pattern_check        -> “friend”
                     | “foe”
                     | “empty”
                     | “this”
                     | id
```