John Conway’s Game of Life

This is a basic implementation of John Conway’s Game of Life.

Ian had mentioned that he thought this would make for a good mathematica project and I think that he’s right. This implementation allows the user to play John Conway’s Game of Life by running the game cell at the end of the notebook and then choosing configurations of “live” squares by clicking on the game board. The user can then see how their chosen configuration evolves by clicking the Start/Stop button. At any point the user can stop the simulation by clicking on the Stop button and can reconfigure the board (Note the board can also be reconfigured while the simulation is in progress). This implementation also comes with 3 pre-set configurations which evolve nicely throughout the course of the simulation.

To play the game skip to the end of the notebook!

The following 5 cells must be run prior to plaing the game (note, they are initialization cells which should been run automatically upon running the game cell at the end of the notebook, if this is skipped, run them manually).

```
InitializeBoard[n_, step_] :=
 DynamicModule[{p},

  If[n > 0,
   A = Table[0, {i, n}, {j, n}];
   buttonlabel = "Start";
   action = "Stop";
   playing = True;
   Glider = {{18, 21}, {19, 22}, {20, 20}, {20, 21}, {20, 22}};
   Exploder = {{18, 20}, {19, 19}, {19, 20}, {19, 21}, {20, 19}, {20, 21}, {21, 20}};
   Tumbler = {{16, 19}, {16, 20}, {16, 22}, {16, 23}, {17, 19}, {17, 20}, {17, 22},
     {17, 23}, {18, 20}, {18, 22}, {19, 18}, {19, 20}, {19, 22}, {19, 24}, {20, 18},
     {20, 20}, {20, 22}, {20, 24}, {21, 18}, {21, 19}, {21, 23}, {21, 24}};

   Print[
    Column[
     {Row[{
        (* Initialize the game buttons *)
    (* Start/Stop button *)
    EventHandler[Dynamic[Button[buttonlabel]],
     {"MouseClicked" :>
       ({buttonlabel = buttonlabel /. {"Start" → "Stop", "Stop" → "Start"},
         action = action /. {"Start" → "Stop", "Stop" → "Start"}})}],
    (* Reset button *)
    Button["Reset",
     {A = Table[0, {i, n}, {j, n}], buttonlabel = "Start", action = "Stop"},
     ImageSize → 71],

    (*Preset configuration button: Glider *)
    If[n == 40, Button["Glider",
      A = Table[0, {i, n}, {j, n}];
      Do[
       A⟦Sequence @@ i⟧ = 1, {i, Glider}]]],

    (*Preset configuration button: Exploder *)
    If[n == 40, Button["Exploder",
      A = Table[0, {i, n}, {j, n}];
      Do[
       A⟦Sequence @@ i⟧ = 1, {i, Exploder}]]],

    (*Preset configuration button: Tumbler *)
    If[n == 40, Button["Tumbler",
      A = Table[0, {i, n}, {j, n}];
      Do[
       A⟦Sequence @@ i⟧ = 1, {i, Tumbler}]]],

    (* Spacing for aesthetic purposes *)
    "          ",

    (* QUIT button *)
    Button["Quit",
     {playing = False, A = Table[0, {i, n}, {j, n}]},
     ImageSize → 71]
   }
  ],

  (* Initialize the game grid *)
  EventHandler[
   EventHandler[
    Dynamic[ArrayPlot[A, Mesh → True, ImageSize → {500, 500}]],
          {"MouseDown" :>
             (p = {-1, 1} * Reverse[Ceiling[MousePosition["Graphics"]]] )},
          PassEventsUp → True],
         {"MouseDown" :> (A[[Sequence @@ p]] = A[[Sequence @@ p]] /. {0 → 1, 1 → 0})}
        ]
       }
      ]
     ],
     (*ELSE*)
     None
    ]
   ]
 (* Find the coordinates of the squares surrounding square {i,j} *)
 SurroundingSquares[{i_, j_}] :=
  DeleteCases[
   Flatten[Table[{i + k, j + l}, {k, {-1, 0, 1}}, {l, {-1, 0, 1}}], 1], {i, j}]
 (* Test whether square at position, pos, will survive the next iteration *)
 LiveOrDie[A_, pos_, n_] :=
  Module[{i, j, Sq, sum, k},

   i = pos[[1]];
   j = pos[[2]];
   Sq = SurroundingSquares[{i, j}] ⋂ Flatten[Table[{i, j}, {i, n}, {j, n}], 1];
   sum = 0;
   For[k = 1, k ≤ Length[Sq], k++, sum += A[[Sequence @@ Sq[[k]]]]];
   Which[
    (*IF*) A[[i, j]] == 1 ∧ (sum == 2 ∨ sum == 3) ,
    True,
    (*ELIF*) A[[i, j]] == 1 ∧ sum ≠ 2 ∧ sum ≠ 3,
    False,
    (*ELIF*) A[[i, j]] == 0 ∧ sum == 3,
    True,
    (*ELIF*) A[[i, j]] == 0 ,
    False
   ]
  ]
```

```
(* Play the game with time step, step *)
PlayGame[n_, step_] := DynamicModule[{B, indices},

  (* Create a set of reference indices *)
  indices = Flatten[Table[{i, j}, {i, n}, {j, n}], 1];

  (* A loop to iterate through game sequences *)
  While[playing == True,
   Which[(*If*)action == "Start",
    (*THEN*)
    B = A;
    Do[
     If[LiveOrDie[A, indices[[i]], n],
      B[[Sequence @@ indices[[i]]]] = 1,
      B[[Sequence @@ indices[[i]]]] = 0], {i, 1, n^2}];
    A = B;
    Pause[step];
   ]
  ]
 ]
(* A function to tie it all together *)
GameOfLife[] :=
  Module[{BoardSize = 40, TimeStep = 0.1},
   InitializeBoard[BoardSize, TimeStep];
   PlayGame[BoardSize, TimeStep]];
```

THE GAME

Running the following cell starts the game (Click “**Yes**” when prompted to evaluate the initialization cells).

The button labelled “**Start**” toggles between “Start” and “Stop”, starting and stopping the simulation, respectively.

The “**Reset**” button stops the simulation and clears the board.

The three subsequent buttons “**Glider**”, “**Exploder**” and “**Tumbler**” are the pre-set configurations, clicking any of them will clear the current game board and load it with their pre-set configuration (Note that these buttons can be clicked at any point during game play, they do not stop the simulation, however they always clear the game board and load it with their pre-set configuration.

The “**Quit**” button clears the game board and ends the cell evaluation.

```
GameOfLife[]
```

buttonlabel | Reset | Glider | Exploder | Tumbler   Quit

```
ArrayPlot[A, Mesh → True, ImageSize → {500, 500}]
```