CSC 4370/6370 WEB PROGRAMMING
Project 3
Due date: 04/21/2023

CSC 4370/6370 WEB PROGRAMMING
Project 3 - Spring
Due date: 04/21/2023

Overview

All groups ◆ Please select your own members ◆ First come basis

Group Requirements: ◆Undergraduates groups - As small as 1 and large as 4

Graduates - As small as 1 and large as 3

This is an excellent opportunity to improve your skills as a *team player*, a highly- desirable type of worker in the real world.

(Read Becoming a Successful Team Member.)

Objective: Applying your knowledge of CSS, PHP-Forms and JavaScript. Provide the

Today◆s Agenda

1. Pick your "OWN" Teamates
2. Choose a leader - liaison to the instructor
3. Brainstorm ideas
4. Plan how you will collaborate/communicate
5. Choose someone to "integrate" the parts done by the team members
6.◆◆ Decide the responsibilities for each team member. e.g.

Please also include in the PPT document the explaination of your TEST CASE for your project. { failure to do so will result in -25 points.}

Requirements

You shall choose one team member as leader for purposes of coordinating the project and reporting to the instructor.◆

PowerPoint slide show MUST include the following:

- ◆ **User** - statement of problem, and general requirements (inputs, outputs, etc.)
- ◆ **Design** - Overview of solution, key design features, user interface, UML class diagrams, pseudo code etc.
- ◆ **Testing** - (if applicable) how tested (e.g., test plan, data used, tracking and reporting bugs, bugs fixed/not fixed, etc.)

Grading

Meet the requirements (see the Requirements section above), you will receive credit Full credit will be based on group that used creativity and whom went above and beyond my requirements.

Your team must turn in the paper as specified above and do the presentations in order to get credit. It will not be sufficient to simply turn in the files to I-College and have it posted on codd. All members post the URL or Project to your student account.

PRESENTATION GRADE SHEET		
	POINTS EARNED	POSSIBLE POINTS
CONTENT		
EXPLANATION OF PROJECT		10
VIEWS		20
CONTROLS/FUNCTIONS		20
MODELS		15
ORGANIZATION		15
USER FRIENDLY-NESS		10
DELIVERY		10

**The leader shall create a separate page that is linked in the PPT to list the roles it should state the following: **

- **Leader's Name**
- **Project Name**
- **Description: a one-sentence description of your project**
- **Roles: who did what**
- **Methodology : << (see the included link) used and a quick summary of how it benefited you on this project**

Topic - Conway's Game of Life

Implement the Game of Life using HTML/CSS, php, Java Script, along with any other libraries

PLAGIARISM DISCLAIMER

We are fully aware of many sources are available and we will check

Acknowledgement should always be made when secondary sources are used. Failure to acknowledge sources will be penalized and any substantial plagiarism for example, unacknowledged quotation that extends for more than a few lines, or extended close paraphrasing of a critical work will automatically be reported to the university disciplinary board and can lead to suspension.

Objective: The game of life is a grid of cells where each cell is in the state of being alive or not. The next generation of the game depends on the current generation and the following rules:

The game consists of a grid of cells, each of which can be alive or dead. For every cycle of the game, the cells can be turned on or off based on the following rules:

1. Any live cell with fewer than two live neighbors ***dies***, which is caused by under population.
2. Any live cell with more than three live neighbors dies, as if by overcrowding.
3. Any live cell with two or three live neighbors lives on to the next generation.
4. Any dead cell with exactly three live neighbors becomes a live cell.
5. If a dead cell has exactly three live neighbors, it comes to
6. If a live cell has less than two live neighbors, it dies
7. If a live cell has more than three live neighbors, it dies
8. If a live cell has two or three live neighbors, it continues living. life - Therefore by repeating the cycle over and over, these simple rules create interesting, often unpredictable patterns.

Please select your members as normal contact me in advance if you have concerns

Design Specifications

1. Create a variable sized table. (Grads use DHTML)
2. The background color of the cells will determine life.
3. The cells can be turned on or off with the mouse.
4. Display the current population (Grads only) and generation.
5. Create a button for each of the following functions (using mouse events such as clicks, drags i.e.):
    A. Start the game
    B. Stop the game
    C. Increment 1 generation
    D. Increment 23 generations
    E. Reset the game (Population=0 or other, Generation=0).
    F. Pattern button/Drop down selection to (Animate selection)
    G. Create a login profile for each user using registration and auto link relocation to the sign-in. *Extra credit if you use some types of login authentication.
        - HTTP basic **authentication**.
        - Form-based **login authentication**.
        - Client certificate **authentication**.
        - Mutual **authentication**.
        - Digest **authentication**.

H. Store the name of players in a file or array and keep track of any user events stats i.e., time and a score etc.

6.🔲 Undergraduates (Please choose 3 patterns --- (1) pattern from the Still life and Two Oscillators, Gliders are welcome and will be considered as a Bonus)

7.🔲🔲 Graduates only - Please create Random Populations (choose 1 pattern from the Still life (2) patterns Oscillators and (1) pattern Gliders).🔲 Graduates only use any demo patterns that will demonstrate the following behavior: {Create any type of animated lit points that will turn off if there are fewer than two or more than three surrounding lit points. An unlit point turns on if there are exactly three lit neighbors}

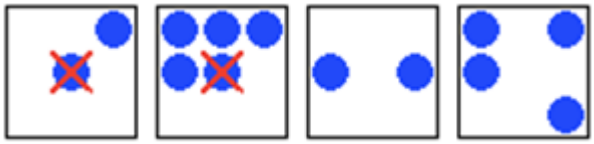
**** Bonus Graduates**** Implement the use of the React.js develop a component that will display the beginning state of the GRID and Ending state of the GIRD (real-time update)

Side note JS react About React component states 🔲 I will continue to post some notes in reference to this frame-work

Component states are what give React its name. Any time a component🔲s state changes, its 🔲render🔲 function is called, updating the output. In essence, each component 🔲reacts🔲 to changes, which is handy for any user interface. Data stored in a state should be information that will be updated by the component🔲s event handlers (changes that should update in real time as the interface is being used).

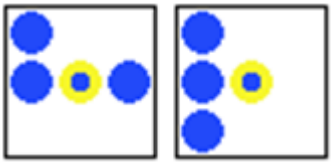
Fundamental Rules

The game is designed around the following simple rules:
1) Any live cell with fewer than two live neighbors dies, as if caused by underpopulation.
2) Any live cell with more than three live neighbors dies, as if by overcrowding.



3) Any live cell with two or three live neighbors🔲 lives on to the next generation.

4) Any dead cell with exactly three live neighbors becomes a live cell.



The operation of the game starts with an initial configuration on a two-dimensional grid. This infinite

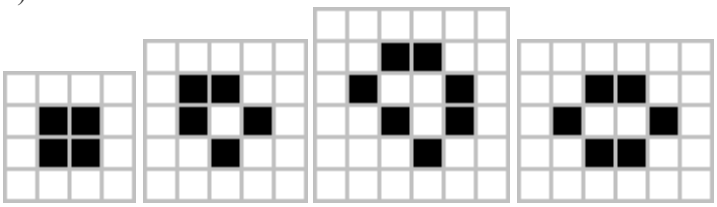
square grid consists of cells with two possible states, alive or dead. Each cell has eight neighbors, namely the eight cells that touch it. The game operates in iterations, called ticks. Each tick applies the four rules of the game to every cell on the board simultaneously.

PATTERNS ARE AS FOLLOWS:

Still Life

These are stable patterns that do not change and can be used to build critical solid parts of complex patterns. These patterns stay in one state which enables them to store information or act as solid bumpers to stop other patterns or keep other unstable patterns stable. Examples of still life include:

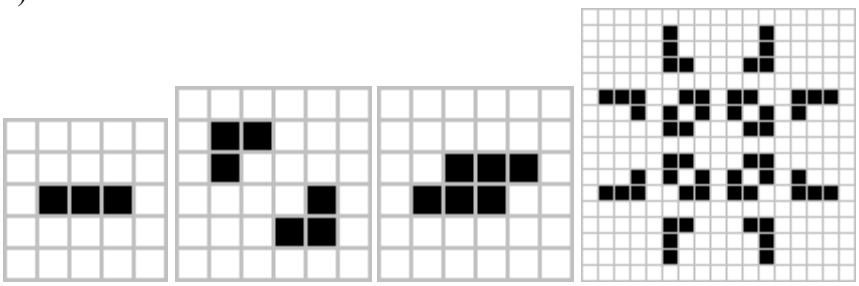
1) **The Block**
2) **The Boat**
3) **The Loaf**
4) **The Beehive**



Block, Boat, Loaf, Beehive (from left to right).

Oscillators
These patterns are more complex and change over a specific number of ticks. They repeat their pattern infinitely. The basic oscillators have periods of two or three ticks, but complex oscillators have been discovered with periods of twenty or more ticks. These oscillators are very useful for setting off other reactions of bumping stable patterns to set off a chain reaction of instability. The most common period-2 oscillators include:

1) The Blinker
2) The Beacon
3) The Toad
4) The Pulsar



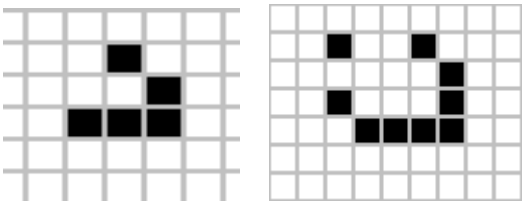
Blinker, Beacon, Toad, Pulsar (from left to right).

Gliders and Spaceships

The spaceship is a pattern that moves, returning to the same configuration but shifted after a finite number of generations. The glider is an example of a simple spaceship and its generations each consist of five live cells. The glider has a period of four and moves diagonally one cell every four generations.

It moves at one-quarter the speed of light.

Other examples of simple spaceships include lightweight, medium weight, and heavyweight spaceships. They each move in a straight line at half the speed of light.

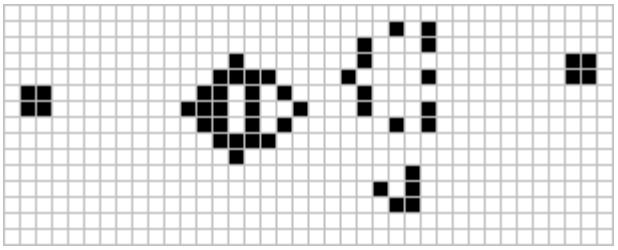


Glider, Lightweight Spaceship (from left to right).

Guns

Guns are repeating patterns which produce a spaceship after a finite number of generations. The simplest gun, called the Gosper glider gun, produces a glider every 30 generations. This fascinating pattern was discovered in 1970 by Bill Gosper. Through careful analysis and experimental testing he developed a pattern which emitted a continuous stream of gliders. Since 1970 researchers and freelance experimenters have discovered hundreds of new patterns and have built thousands of intricate machines and devices using these and other simple parts.

Theoretically the many different possibilities of these four simple rules allow the development of any kind of computing. A Turing machine has been implemented in Conway's Game of Life. There are hundreds of other amazing patterns.



Gosper Glider Gun.

Design requirements

A. Grid system
B. Awesome Layout designs just beautiful.
C. Colors - Fonts and texts
D. Links and navigation
E. Images / Icons
F. Forms and buttons
G. Validation
H. Responsive Web Design
I. Style Guide and component approach
J. Delivery Analysis and pre-work phases

Remember these guidelines for a proper design

- Ensure the functions are all working logically
- Ensure proper access by everyone regardless of user.
- Use valid CSS and follow CSS best practices.

Additional Resources:

🛠️ Bootstrap Grid System (v4)

🛠️ Flexbox Grid

📖 Don't Overthink It Grids | CSS-Tricks

SUBMISSION REQUIREMENT

Create a YouTube Video: Name of the task Final Project 03_TeamName

- Provide a description: one- minute introductions description of your project and group members
- This video must range for 10 - 15 minutes in presenting which will include the demo run
- and code snippets of your project.
- You must be submitted on I-College by the due date and time. Late homework will not be graded, as stated in the course grading policy keep in mind the last day is may 4th and you will have a Final Project.
- No email or hard copies of homework will be accepted by me (submit to drop-box).
- Every team member must participate in this video if a member is missing that member will Earn a grade of zero.
- Be sure to show several clips during the life cycle development failure will result in deduction of points)
- Use the best suited recording method. Make sure the voice is clear.
- Create a channel at YouTube and name it as your group name
- Once ready, upload the video to your channel.
- Include the link to this channel with your submission and you will incorporate the video for the core of your PowerPoint presentation.
- Log into (ICollege), select the class drop box folder, please be sure to select the correct
- folder for the given assignment and upload the file there.