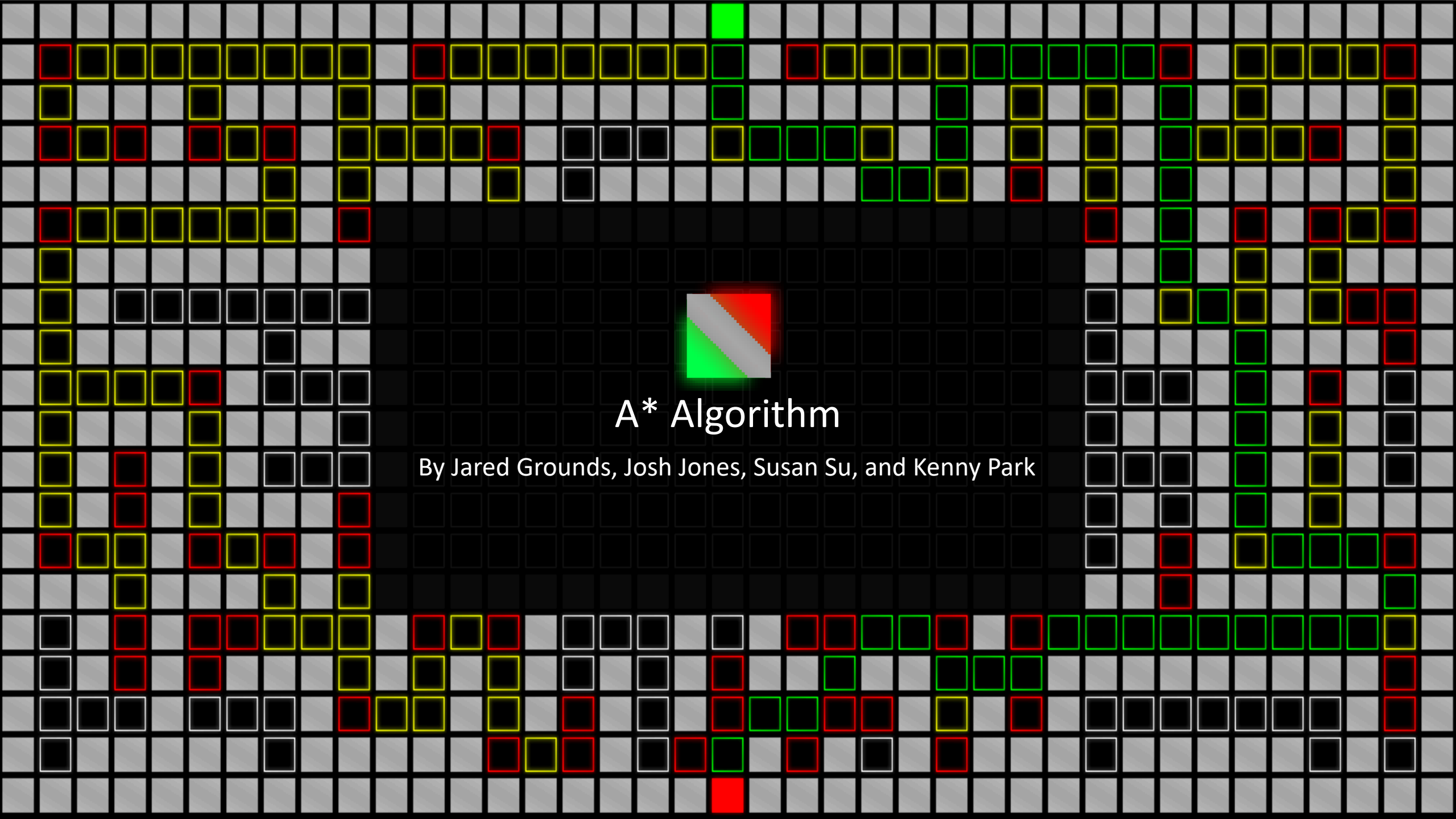
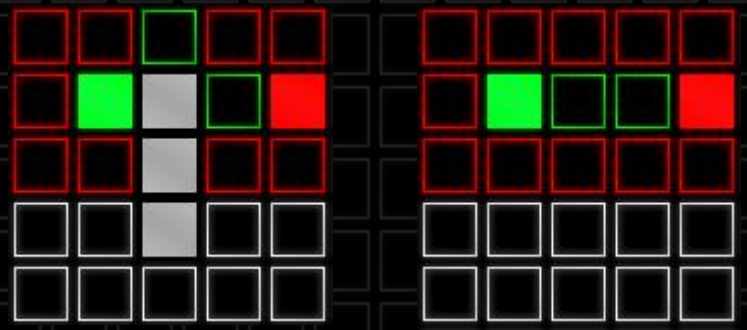


A* Algorithm

By Jared Grounds, Josh Jones, Susan Su, and Kenny Park

Problem Description

- Pathfinding AI is becoming more prominent
  - Real-time navigation
  - Mathematics
  - Video games
- It is necessary to visualize the logic behind such search algorithms



How can we determine the shortest path from one point to another while avoiding obstacles?

Problem Solution

A* Search Algorithm

- Peter Hart, Bertram Raphael, Nils Nilsson (1968)
- Highly popularized search algorithm
- Finds the shortest path from a start point to a destination point while avoiding obstacles
- Utilizes "heuristics" while searching for the shortest path

Programming a visualizer

- Implements A* Search Algorithm
- Visual C++ Forms

Data Structures – Tile Class

- Derives from C++/CLI PictureBox component

Properties:

- Integers for hCost, gCost, hCost, tile type, and XY positions on the tile grid
- ArrayList of valid neighboring tiles
- Parent tile that a given tile is selected from
- Boolean for if the tile type can be changed

Methods:

- Various get/set functions for tile properties
- Click and hover Event-handlers for when a tile is interacted with on the UI

Data Structures - User Input

Build Mode

Void tileClick(){

If MouseButtons:: Left

- Switch upon click
  - Case 0 set the tile Type to 1 (green, start)
  - Case 1 set the tile Type to 2 (red, end)
  - Case 2 set the tile Type to 0 (black, empty)
  - Default set the tile Type to 0 (black, empty)

- Else If MouseButtons:: Right
  - If isDragging is true, set the hoveredTIle to Type 3 (grey, obstacle)

}

Data Structures – Exception Handling

- Void loadButton_Click(){
  - Show load dialog
  - Try parsing the file into the grid
  - Catch any errors

  }

- Void saveButton_Click(){
  - Show save dialog
  - Try parsing the grid into a text file
  - Catch any errors

  }

Data Structures – A* Algorithm

Void runAlgorithm(){

ArrayList^ openTiles

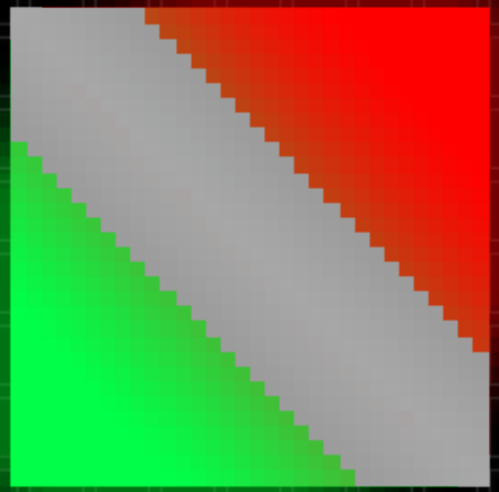
ArrayList^ closedTiles

- Set the starting tile to the first index in openTiles

While openTiles is NOT empty

- Set current tile to the tile in openTiles with lowest fCost
- Change current tile from openTiles to closedTiles
- Check if current tile is the end tile, if it is then a valid path has been found
- If a given neighboring tile of the current tile is in closedTiles or is an obstacle, skip it
- If the path to the neighboring tile is shorter or neighbor is in closedTiles, calculate its fCost and set the parent to the current tile
- If the neighboring tile is not currently in openTiles, add it

}



A* Algorithm Demonstration

Notable Observations

- Instantiating a new variable in every iteration of a for-loop

```
for(int i = 0; i < tiles->Count; i++){
Tile^ currentTile = (Tile^)tiles[i];
}
```

```
Tile^ currentTile;
int tileCount = tiles->Count;
for(int i = 0; i < tileCount; i++){
currentTile = (Tile^)tiles[i];
}
```

- Reading an image from file while instantiating grid, rather than initializing an image variable.
- Certain C++/CLI event-handlers conflict with each other.
- Switched from Pythagorean theorem for hCost (distance from current tile to end tile)
  - Less resource intensive
  - Gave shorter distance path

Future Work

- Allow option to configure grid size and heuristic cost formula
- Convert to Unity3D C#, or a similar engine, for practical usage
- Implement more algorithms
  - BFS can be faster in certain situations (poor heuristic)
  - Dijkstra's algorithm for weighted tiles/paths