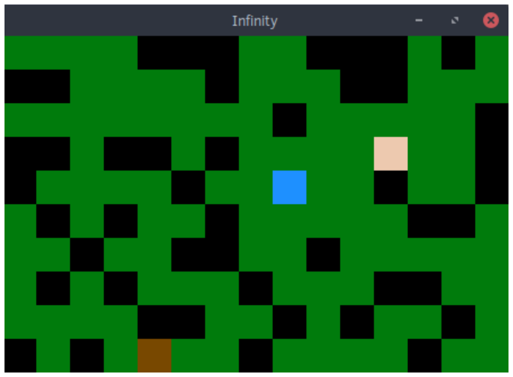
Infinity

Group: Josiah Witt, Sam Nguon, and Jose Chan

Description



Infinity is a game that allows a player to move around in a randomly generated 2D world that infinitely scrolls to the right. The arrow keys control the player (in blue), which can only move on the floor (grass - green, sand - tan, dirt - brown). A mouse click creates or removes walls (in black), while a mouse drag moves walls. The description of classes below outlines what we have as the core functionality and are implemented in the attached code.

Classes

- **GameBoard:** has-a Block. Has a hold of every block object in the game.

  Enums/Structs/Consts:
  - `enum GameDirection {DIR_UP, DIR_DOWN, DIR_LEFT, DIR_RIGHT}`
  - `const int GAME_VERSION`

  Fields:
  - `int numBlocksWide`
  - `int numBlocksHigh`
  - `int blockWidth`
  - `int blockHeight`
  - `vector<vector<Block>> blocks`
  - `Player player`
  - `int seed`
  - `mt19937 rand`
  - `double percentWall`
  - `vector<vector<shared_ptr<Block>>> board`
  - `map<int, map<int, shared_ptr<Block>>> changes`
  - `int leftDisplayEdge`
  - `uniform_real_distribution<> dist`
  - `string gameFilename`

  Methods:
  - `void save(string filename)`
  - `void load(string filename)`
  - `int getGamePixelWidth()`
  - `int getGamePixelHeight()`
  - `int getNumBlocksWide()`
  - `int getNumBlocksHigh()`

- `int getBlockWidth()`
- `int getBlockHeight()`
- `int convertVectorXToPixelX(int vectorX)`
- `int convertVectorYToPixelY(int vectorY)`
- `int convertPixelXToVectorX(int pixelX)`
- `int convertPixelYToVectorY(int pixelY)`
- `void movePlayer(int amountX, int amountY)`
- `void swapPlayerColor()`
- `void changeFloorTypeUnderPlayer(FloorType f)`
- `bool moveWall(int lastX, int lastY, int currentX, int currentY)`
- `bool addWall(int pixelX, int pixelY)`
- `bool removeWall(int pixelX, int pixelY)`
- `void generateBoard(int seed, Player player)`
- `void generateColumn()`

● **Block**: Abstract class. The cornerstone of this game, just about everything built on the screen is made of blocks.

Enums/Structs/Consts:
- `struct Color {double r, g, b}`
- `enum BlockType {PlayerBlock, FloorBlock, WallBlock}`

Fields:
- `Color color`

Methods
- `Color getColor() const`
- `void setColor(Color c)`
- `void draw(int pixelX, int pixelY, int width, int height) const`
- `virtual bool canMoveOnTop() const = 0`
- `virtual BlockType getBlockType() const = 0`
- `virtual json toJson() const`
- `virtual void fromJson(json j)`

● **Player:** is-a Block. The main controllable aspect of the game, being able to move around and change colors.

Fields:
- `Color alternateColor`
- `int vectorX`
- `int vectorY`

Methods:
- `virtual bool canMoveOnTop() const override`
- `virtual BlockType getBlockType() const override`
- `Color getAlternateColor() const`
- `void setAlternateColor(Color c)`

- `void swapColor()`
- `void setVectorX(int x)`
- `void setVectorY(int y)`
- `int getVectorX() const`
- `int getVectorY() const`

- **Wall:** is-a Block. The obstacles and general boundaries of the game.

  Fields:
  - `// The wall currently has no fields`

  Methods:
  - `virtual bool canMoveOnTop() const override`
  - `virtual BlockType getBlockType() const override`

- **Floor:** is-a Block. Type of blocks that the Player can walk on.

  Enums/Structs/Consts:
  - `enum FloorType { GrassFloor, SandFloor, DirtFloor }`

  Fields:
  - `FloorType floor`

  Methods:
  - `virtual bool canMoveOnTop() const override`
  - `virtual BlockType getBlockType() const override`
  - `FloorType getFloorType()`
  - `void setFloorType(FloorType f)`