***Class Game:***

The public member functions in the class Game are all non virtual because there are no derived classes of Game.

Void play() is the function where everything runs. Here the random pieces are generated and after each piece becomes stationary, it determines whether the player has successfully finished a level or not.

Bool playOneLevel() checks if the number rows left is zero. If so, it empties the tank and returns true.

Void displayPrompt(std::string) is a helper function that gets called whenever a prompt needs to be displayed.

Void displayStatus(): this is called once whenever a pieces comes to an rest and a new piece is created. This shows the player all the information about the game including next piece, score, level and rows left,

Int level() returns the current level. This is a helper function.

~Game() because there is dynamically allocated element in game, we need destructor.

***Class Tank:***

int display(Screen& screen, int x, int y, int level): This function is called once and lasts the duration of the falling of a piece. Here is where the program displays the piece, controls the speed at which the piece is falling and also executes all the commands that the player inputs while the piece is falling(rotate, move right, move left, etc.). All the special affects that pieces have are also implemented here.

void preDisplay(Screen& screen, int x, int y); this function displays empty spaces at the old positions of a piece before the piece is displayed at a new position.

void changeTank(int x, int y); this function is called when a piece stops falling. It changes 2D vector of the tank so that the piece becomes part of the tank.

void displayTank(Screen& screen, int x, int y); this simply displays the tank and nothing more. This is a helper function for int display.

Tank (int tankWidth, int tankHeight); This is the constructor for Tank. It sets up the 2D array so that it the borders of the tank is shown as ‘@’ and the rest is filled with spaces.

bool insert(Piece\* newPiece); this function puts the nextpiece of the game into the tank so that the tank can start controlling it and make it fall/move.

bool goodToGo(std::vector<std::vector<char>> shape, int x, int y) const; this function is called just before a piece is about to move. It checks if the position the piece is about to move into contains no walls/other pieces and prevents the piece from moving if the new position is not legal.

const Piece\* showCurrent() const; This is a helper function that tells the called what the current piece in the tank is. This returns a const pointer and is a const function because calling this funtion is not supposed to change anything.

int clearRow(); this is called when a piece comes to a rest and checks if the tank now contains any rows that are full. If so, it empties the full rows and shifts everything above down and returns the number of rows eliminated.

void clearTank(); This clears the whole tank except the border. Called at the end of a level.

void fillFoam(int x, int y, int left, int right, int top, int bottom); This is called when a FOAM piece comes to an stop and it fills the surrounding positions with ‘\*’ according to the rules.

~Tank() Because there is dynamically allocated element, we need a destructor

***Class Piece:***

virtual Piece\* rotate(); this is virtual because the way my rotation mechanism works is that the current piece in tank is deleted and replaced by a new piece of the same type but is constructed in different ways. Therefore each piece needs to have its own rotate function. This is not pure virtual because some of the pieces – namely the O piece, Vapor piece, and the Foam piece, don’t need a separate rotation function because they don’t rotate at all. So for those pieces, they share the same useless definition of rotate(). For the rest of the pieces, this function constructs a new piece using desired configuration and returns a pointer to that created piece so that tank can replace its m\_current pointer.

void display(Screen& screen, int x, int y) const; This function displays the 2D array in the piece element at the position provided. This function is called by tank display repeatedly within the duration of the falling process.

const std::vector<std::vector<char>> showShape() const; This function returns the 2D vector array. This is a helper function and so shouldn’t change anything. Therefore it’s a const function with const return type.

virtual std::string type() = 0;

This is another helper function that tells the caller what the type of a piece is. This is pure virtual because each piece has a different name.

int showStatus() const; this function tells the caller when orientation the piece is in (most pieces have 4 different orientations, some have two)

virtual ~Piece(); the destructor is virtual because we have derived classes.

Each individual Piece derived class has a virtual destructor, a virtual type() function and (except O, VAPOR, FOAM) a virtual rotate function().

2. There was nothing that was purposefully left out.

3.

a) When the crazy piece rotates, (because of its diagonal shape), it is possible for part of it to rotate into a grid that is normally not reachable by other pieces without overlapping with anything. I just assumed that it is legal to do so.

b) When the vapor bomb deploys within the first two rows of the tank, then the foam might reach out side of the tank. I assumed that it is ok to first fill as much space within the tank as the vapor bomb can and continue with the game until it terminates in regular ways.