

* **Main screen is like chess:** 
  + Login window with 2 options:
    - 1. Start
    - 2. High score
  + Clicking “start” will open game window
  + Clicking “high” score will open high score window
* **Starting game window:**
  + Read the map from txt file
  + 28 by 31 grid. Each cell in the grid is an image. Can be:
    - 1. Pacman
    - 2. Ghost
    - 3. Wall
    - 4. Dot
    - 5. Superpower
    - 6. Nothing
  + UI stuff:
    - Show high score on top
    - Show lives and level at the bottom
  + Countdown before the game begins (3,2,1, start or something like that)
* **Update map every timeout signal**
  + Check if a new key was pressed since the previous timeout signal
  + Change location of pacman accordingly
  + Change location of ghosts
  + Check if pacman eats a dot (not including ghost yet)
  + See if score gets updated
  + Check if level finished
  + If (not superpower):
    - Check if pacman is eaten
    - If (eaten) :
      * Minus 1 life
      * If (no more lives): show game over screen
      * Else: Return to starting position
  + Else if (superpower):
    - Check if pacman eats a ghost
    - If (eats a ghost):
      * Increase score
      * Put eaten ghost back to starting position
      * Increase number of ghosts eaten variable
    - Increment superpower timer
  + If (there is ghost in middle box): increment timer

**This is just a rough sketch and I sort of just looked at the stuff that they had for chess and pipes. Didn’t really include the complicated stuff like the ghost movement yet.**

**Classes:**

**1. Main window:**

Data members:

* game\_window\*

Member functions:

* Constructor, destructor
* start\_button\_clicked\_handler()
* game\_window\_closed\_handler()

**2. Game window:**

Data members:

* Square[31][28]
* (int) high\_score
* pacman\* pacman
* ghost\* ghost1
* ghost\* ghost2
* ghost\* ghost3
* ghost\* ghost4
* (int) score
* (int) lives
* (int) level
* Qtimer

Member functions:

* Constructor, destructor
* load\_map()
* load\_high\_score()
* refresh\_frame() → this is the slot that the timer connects to
* quit() → a slot for quitting
* game\_over() → a signal for game over
* process\_user\_input() → the thing that takes in keyboard inputs (this updates current pacman direction)
* is\_level\_finished()
* exists\_ghost\_in\_box()
* update\_score() → updates LCD display
* update\_map() → updates the gui by calling functions from the square class
* back\_to\_starting\_pos() → call this when pacman dies

**3. Square:**

* character\* piece
* set\_image() → calls get\_image() from the piece character\* then puts the appropriate image
* Row, col, width, height → not really sure what these are for but the chess had it so i’m guessing it’s for the UI

**4. Character (abstract base class)**

* Constructor, destructor
* Character\* grid[31][28] → pointer to the whole map
* get\_image() → virtual
* Int row, col
* getRow(), getCol()

**5. Wall (inherits character)**

* get\_image()

**6. Dot (inherits character)**

* get\_image()
* (int) points

**7. Pacman (inherits character)**

Data members:

* (bool) has\_eaten\_piece
* (int) superpower → this tells us how much time is left. If currently not in superpower mode, this is set to -1
* (int) direction

Member functions:

* Constructor, destructor
* get\_image()
* has\_eaten\_piece() → for updating high score
* get\_superpower() → returns superpower
* update\_direction()
* move()

**8. Ghost (inherits character)**

Data members:

* (int) points → 200, 400, 800, 1600. Update\_points() is called whenever another ghost is eaten.
* (int) time\_in\_box → indicates the time left before ghost can exit the box. When the
* (bool) is\_eaten

Member functions:

* Constructor, destructor
* get\_image()
* get\_time\_in\_box()
* eaten() → updates is\_eaten
* set\_color(bool eaten)
* move()
* update\_points()