OUTLINE DOCUMENT 9.3H

Please download zip file of 9.3H code from Google Drive through this link:
< https://drive.google.com/open?id=1Be9gd8S1HB6rvi-joghWYT4SEkmxjHgf >

I/ About program:

This program is a small game, based on idea of 2048. This game is called Fruit 2048

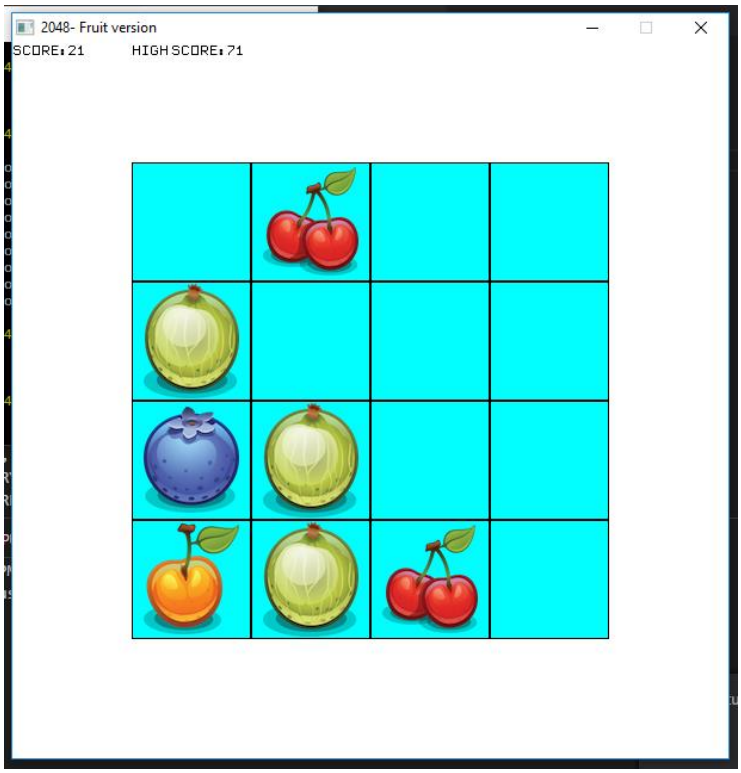
How to play: Use arrow keys to move the images of fruit. When to images with the same fruit touch, they merge into one and you will get point for that.

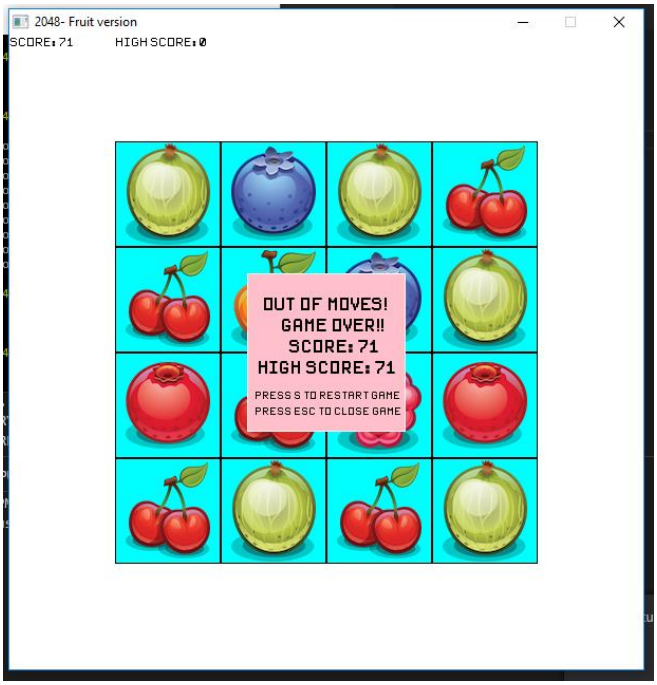
In the early version, try to get an image of melon and you will win the game

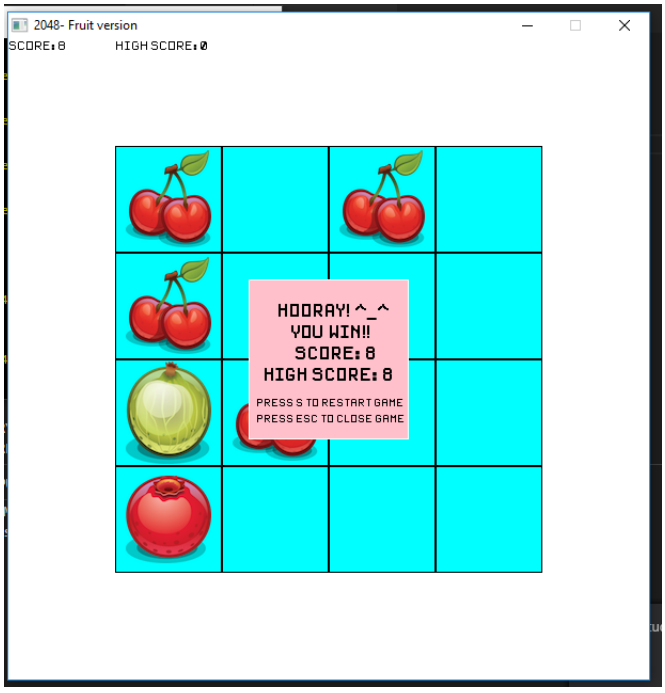
II/ Main components:

- Blocks of image of fruit:
  - There are 13 different types of fruit, in order:
    - CHERRY,
    - GOOSEBERRY,
    - BLUEBERRY,
    - POMEGRANATE,
    - APRICOT,
    - RASPBERRY,
    - BLACKBERRY,
    - STRAWBERRY,
    - CURRANT,
    - APPLE,
    - ORANGE,
    - LEMON,
    - MELON
- Score:
  - The score of current game
  - Update whenever 2 images of fruit be merged
- High Score:
  - Best score for previous games
  - Update when finish one term of game
- Win:
  - User will win when the lasted fruit appears (Melon)
  - Score will be displayed and compare to high score to update
- Lose:
  - Game over when there are no more movements
  - Score will be displayed and compare to high score to update

*** The game can be play again many times. Press S to start new game or press Esc to close window


2048- Fruit version
SCORE: 21 HIGH SCORE: 71


2048- Fruit version
SCORE: 71 HIGH SCORE: 0
OUT OF MOVES!
GAME OVER!!
SCORE: 71
HIGH SCORE: 71
PRESS S TO RESTART GAME
PRESS ESC TO CLOSE GAME


2048- Fruit version
SCORE: 8 HIGH SCORE: 0
HOORAY! ^_^
YOU WIN!!
SCORE: 8
HIGH SCORE: 8
PRESS S TO RESTART GAME
PRESS ESC TO CLOSE GAME

Note: The Win picture is just example to show how it will be displayed. User have to get the Melon (the 13th fruit) to win the game

III/ Explanation some main procedures, functions and some code line:

1/ 2-dimension array

This game uses the 2-dimension array with size 4*4 to store and display the images of fruit

- Declaration:

```
int fruits[4][4];
```

- Access array: Using 2 loops to access each elements of array

```
    for (int i=0; i< SIZE; i++)
    {
        for (int j=0; j< SIZE; j++)
        {
            // Code block to execute
            // element fruit[i][j]
        }
    }
```

** Value in array: fruit[i][j]

0: no images for this box

1-13: contains fruit for this box, number from 1 to 13 will represent the types of fruit

2/ In program.cpp

```
12
13  int main()
14  {
15      open_window("2048- Fruit version",600,600);
16      load_resources();
17
18      fruit_game_data game;
19      game = new_game();
20      int high_score = 0;
21      while (not quit_requested())
22      {
23          process_events();
24          check_quit_game(game);
25          if ((game.win) || (game.lose))
26          {
27              draw_finish_box(game.win,game.lose,game.score,high_score);
28              if (key_typed(ESCAPE_KEY)) break;
29              if (key_typed(S_KEY))  game= new_game();
30
31
32          }
33          else
34          {
35              handle_input(game);
36              draw_game(game,high_score);
37          }
38
39      }
40      return 0;
41  }
```

- (not quit_requested()) and (process_events()) used to keep the window open till user want to quit it
- The high_score is not declared inside game as this value will use for many terms of game and has to update after each game
- Use break to escape from the loop when user want to quit game so that the window will be turn off

```
            if (key_typed(ESCAPE_KEY)) break;
```

- If user want to start new game (press S key), game will be assigned as new_game() – assigned as default values

```
            if (key_typed(S_KEY))  game= new_game();
```

3/ fruit.cpp

```
/**
 * Procedure use 2 loop to draw boxes of fruit
 * @params
 *          game        current playing game. Pass by reference
 *          x, y        position to draw first bitmap, change through loops
**/
void draw_box(const fruit_game_data &game, double x, double y)
{
    bitmap to_draw;
    double xx,yy;
    xx = x;
    yy = y;
    for (int i=0; i< SIZE; i++)
    {
        for (int j=0; j< SIZE; j++)
        {
            draw_fruit(game.fruits[i][j],xx,yy);
            xx = xx + BOX_SIZE;
        }
        yy = yy + BOX_SIZE;
        xx = x;
    }
}
```

- Example for using 2 loops to access each elements of array, in this case: access to draw images of fruit
- Images will be draw in rows, from rows 0 to rows 3; each row draw from column 0 to column 3
- Variable xx, yy:
    - store the values for position of each elements
    - Update for each time of loop

```
/**
 * Procedure use 2 loop to draw boxes of fruit
 * @params
 *          game        current playing game, Pass by reference
 *          count       number of current fruits
**/
void random_add_fruit(fruit_game_data &game, int count)
{
    int tmp, tmp_current;
    if ((SIZE*SIZE-count-1)>0)
        tmp = rnd(SIZE*SIZE -count-1)+1;
    else tmp=1;
    tmp_current=0;
    for (int i=0; i< SIZE; i++)
    {
        for (int j=0; j< SIZE; j++)
        {
            if (game.fruits[i][j]==0)
            {
                tmp_current +=1;
                if (tmp_current == tmp)
                {
                    game.fruits[i][j]=1;
                    return;
                }
            }
        }
    }
}
```

- Each time user press arrow key to move images, one new fruit (Cherry) will be added into game.
- Count: the number of current fruits of game (not include new fruit)
- From Count, we can find the number of left boxes, random to choose which box will be added new Cherry, then use for-loop to access that box.

4/ Fruit_game.cpp

- To make the program short, we just call one procedure for 2 actions (Left and Up arrow key); similarly one procedure for 2 actions (Right and Down arrow key).
- Reason:
    - Left and Up:
        - Loops of both actions run from the smallest index number to the highest
        - Different:
            - Left arrow key: Loop for columns is inside loop for rows

        - Up arrow key: Loop for rows is inside loop for columns
    - Right and Down:
        - Loops of both run from the smallest index number to the highest index number for first loop and from the highest to the smallest for the second loop
        - Different
            - Right arrow key: Loop for columns is inside loop for rows
            - Down arrow key: Loop for rows is inside loop for columns

- Thus, we have

```
- void presskeyLeftUp_call(fruit_game_data &game, char key)
- void  Update_array_LeftUp(int fruits[][SIZE],int i,const vector <int>
  &tmp,char key)
```

```
- void presskeyRightDown_call(fruit_game_data &game, char key)
- void  Update_array_RightDown(int fruits[][SIZE],int i,const vector <int>
  &tmp,char key)
```

a/ Function and procedure can be reused by all actions:

- All actions are used vector to store and check for the same images
- Reset vector before second loop
    - `reset_vector`

- Add value for array; return value from array
    - `add_value`
    - `assign_value`
- Merge 2 same images
    - `merge_2_fruits`

IV/ Core struct:

```
struct fruit_game_data
{
    int fruits[4][4];
    int score;
    bool win;
    bool lose;
};
```

- fruit_game_data:
    - fruit[4][4]: 2 dimension array sized 4*4 , store number which represents type of fruit
    - score: score of current game

- win: Boolean variable to check whether user win
- lose: Boolean variable to check whether user lose

V/ Procedures and function:

1/ fruit.cpp

```
//fruit.cpp
bitmap fruit_bitmap(fruit_kind kind);
void draw_box(const fruit_game_data &game, double x, double y);
void draw_fruit(int value, double xx, double yy);
void random_add_fruit(fruit_game_data &game, int count);
```

- fruit_bitmap:
  - return the required fruit bitmap
- draw_box:
  - use 2 loops to call procedure of draw fruit for each element of array
- draw_fruit:
  - draw fruit
- random_add_fruit:
  - random add fruit (cherry) in one of left boxes of game

2/fruit_game.cpp

```
//fruit_game.cpp
fruit_game_data new_game();
void draw_game(const fruit_game_data &game, int high_score);

double calculate(string s);
void draw_line(string s1, string s2, string s3,string s4, double x, double y);
void draw_finish_box(bool win, bool lose, int score, int &high_score);
void check_quit_game(fruit_game_data &game);

void  merge_2_fruits(int &tmp_value,vector <int> &tmp, int &score);
void  Update_array_LeftUp(int fruits[][SIZE],int i,const vector <int> &tmp,char key);
void  Update_array_RightDown(int fruits[][SIZE],int i,const vector <int> &tmp,char key);

void handle_input(fruit_game_data &game);

void reset_vector(vector<int> &tmp);
void add_value(int fruits[][SIZE],int i, int j, int value, char key);
int assign_value(int fruits[][SIZE],int i, int j,  char key);
void presskeyLeftUp_call(fruit_game_data &game, char key);
void presskeyRightDown_call(fruit_game_data &game, char key);
```

- new_game
  - Return data of new game

    - Assign all variable as default values
- draw_game
    - draw game frame, boxes of fruit, score and high score
- handle_input
    - Receive user input, based on typed key, do specific actions
- presskeyLeftUp_call
    - Execute code block when user press Left arrow key or Up arrow key
    - Use vector to check same images
    - Call procedure to put back values into array
- presskeyRightDown_call
    - Execute code block when user press Right arrow key or Down arrow key
    - Use vector to check same images
    - Call procedure to put back values into array
- reset_vector
    - remove all values (if have) inside vector
- add_value
    - Add value to element of array
- assign_value
    - return value from element of array
- merge_2_fruits
    - Merge 2 fruits into 1
- Update_array_LeftUp
    - Update array, using for Left and Up actions
- Update_array_RightDown
    - Update array, using for Right and Down actions
- check_quit_game
    - Check whether game quit or not
        - If user get the Melon => game win
        - Check if there are any left boxes and check there are any moves left
            - If no left boxes and no more movements => game over
- draw_finish_box
    - Draw box when finish game
    - Displayed score, ask for new game or quit game
- draw_line
    - Draw some lines of text on the screen
- Calculate
    - Calculate position to display text

***** While Playing a game, if you find any error or problem, please send e-mail to this address: ledd1703@learning.deakincollege.edu.au (Amie)**

Thank you for your help.