PROJECT REPORT ON

**2048 GAME**

(UNDER THE PARTIAL FULFILLMENT OF THE

UNIVERSITY FOR COURSE OF T.Y.BSC

COMPUTER SCIENCE)

SUBMITTED BY:

**MS. SAKSHI RAJESH GUPTA**

GUIDED BY:

**MS. BHUMIKA NAKUM**

DEPARTMENT OF COMPUTER SCIENCE

PARLE TILAK VIDYALAYA ASSOCIATION’S

MULUND COLLEGE OF COMMERCE S.N.ROAD,

MULUND (WEST), MUMBAI-80

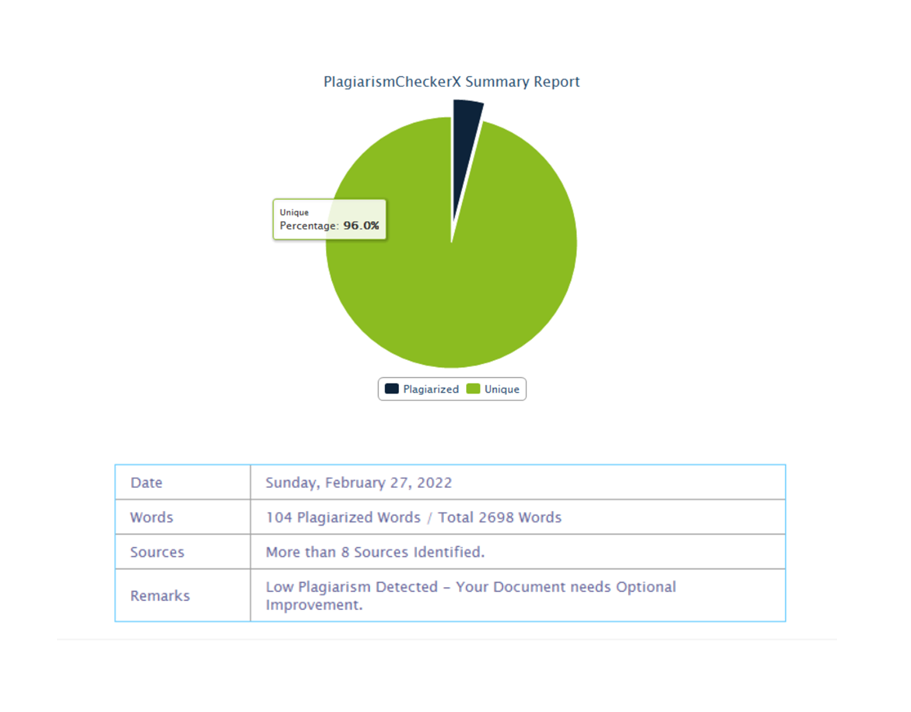
UNIVERSITY OF MUMBAI 2021-2022

**PLAGIARISM REPORT**



Plagiarism Checker X Originality Report

**Similarity Found: 4%**



ACKNOWLEDGEMENT

I have a great pleasure in representing this project report entitled “2048 GAME” and I grab this opportunity to convey my immense regards towards all the distinguished people who have their valuable contribution in the hour of need.

I like to extend my gratitude to our beloved Principal Dr. Sonali Pednekarfor her timely and prestigious guidance.

I take this opportunity to thankMs. Reena Shah, Coordinator of the Department and all the professors of the Department of Computer Science of Mulund College Of Commerce, for giving me an opportunity to complete this project and the most needed guidance throughout the duration of the Programme.

I am extremely grateful to my project guide Ms. Bhumika Nakum her valuable guidance and necessary support during each phase of the project. She was the source of continuous encouragement as each milestone was crossed.

A special thanks to the University Of Mumbai for having prescribed this project work to me as a part of the academic requirement in the Final year of Bachelor of Science in Computer Science.

Finally, I also owe to my fellow friends who have been a constant source of help to solve the problems that cropped up during the project development process.

Sakshi Rajesh Gupta

**INDEX**

|  |  |  |
| --- | --- | --- |
| **SR. NO.** | **TOPIC** | **PAGE NO.** |
| **1.** | **TITLE** | **1** |
| **2.** | **INTRODUCTION** | **2** |
| **3.** | **REQUIREMENT SPECIFICATION** | **4** |
| **4.** | **SYSTEM DESIGN DETAILS** | **5** |
| **5.** | **SYSTEM IMPLEMENTATION** | **21** |
| **6.** | **RESULTS** | **43** |
| **7.** | **CONCLUSION AND FUTURE ENHANCEMENTS** | **48** |
| **8.** | **REFERENCES** | **49** |
| **9.** | **ANNEXURE** | **50** |

1. **Title : 2048 Game**

1. **Introduction**

**2.1. Objective of the project**

* 2048 presents with a 4×4 grid. When you start the game, there will be two “tiles” on the grid, each displaying the number 2 or 4.
* Your objective is to reach 2048 before the board fills up. The game is won when a tile with a value of 2048 appears on the board. Players can continue beyond that to reach higher scores. When the player has no legal moves (there are no empty spaces and no adjacent tiles with the same value), the game ends
* It helps to increase concentration of the user. To increase thinking ability of the user.

To increase time management ability of the user.

* 1. **Description of Current System**
* The current system, gaming allows cyber criminals to manipulate conversations. They may single out your child in a general chat channel and then start sending personal messages that ask for detailed personal information and can misuse them.
* The current system have many forms and tricks. Some games use the "freemium" model, which means they give you some content for free, however, for full game features, functions and access payment is required. These games require users to attach a credit card to their gaming profile. Their card is automatically charged whenever users purchase new items or services.
* In the current system, when too many users playing game at the same time they face loading error, server error, connection error etc. even the data of the users are interchange.
* The major cost of current system gaming comes from the charges of internet connection. The internet Service Provider (ISP) may impose large data charges depending on the time the user spends for gaming.
  1. **Limitation of Current System**
* In online games they have various deficiencies.
* Users cannot play when there is no empty spaces and no adjacent tiles with the values.
* They do not have undo button so the users can correct their mistake.
* Users cannot play the game when there is no Internet connection
  1. **Description of Purposed System**
* The purpose of this game is design user-friendly.
* Data can be stored in word format.
* The rule is that it remains in play until the player has no legal moves.
  1. **Advantage of Purposed System**
* User friendly
* Accurate
* Speed
* Easy to play
* Less time consumption
* Store data
* Teaches time management

1. **Requirements**

**3.1 Software requirements**

* Operating system used: Windows 10.
* Python IDLE

**3.2 Hardware requirements**

* **Processor speed:-** 1.5 GHz
* **RAM:-** 4GB
* **Disc Space:-** 1GB and less
* **Front end used:-** Python
* **Back end used:-** Python, MySQL

**3.3 Data Requirements**

* Whether we have won or loss and score and best score are visible on main page.
* we can save the game and open it again to continue the game.
* The game is store in the word format.

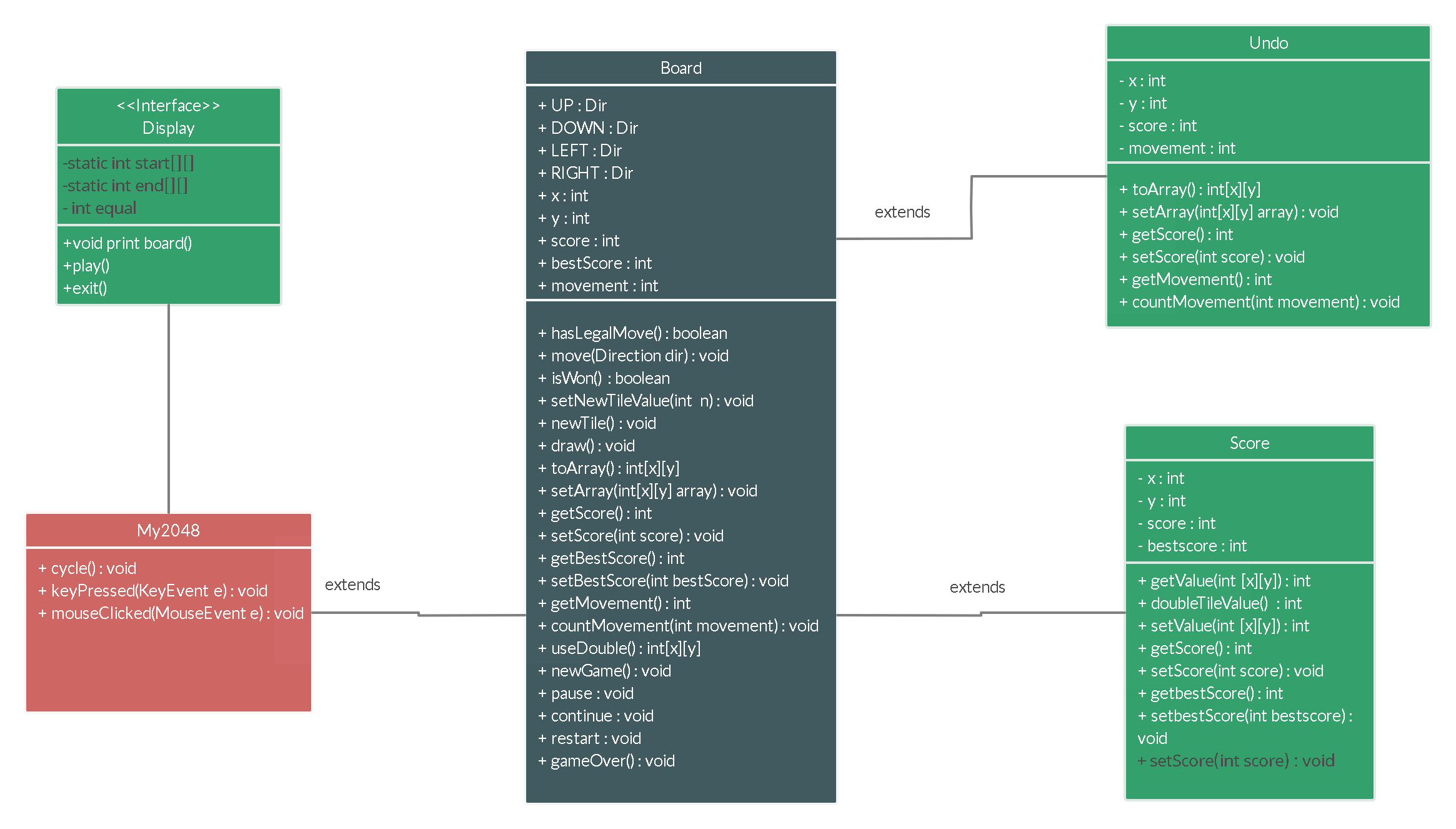
**3.4 Fact Finding Questions**

* What is the Purpose of 2048 Game?
* Why do you want 2048 Game?
* When will be the 2048 Game be ready?

**4. System Design Details**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Sr.no** | **Events** | **Trigger** | **Source** | **Activity** | **Response** | **Destination** |
| 1 | Restart | btnRestart | User | Restart Game | Redirect to new game | Main Page |
| 2 | Save | GameOption2 | User | Automatically saves data | Successfully  Game Save | Main Page |
| 3 | Open | GameOption3 | User | Automatically  Load last data | Open Saved game data | Load Main Page |
| 4 | Undo | btnUndo | User | Undo move | Undo moves in the game | Main Page |

**4.2 Class Diagram**

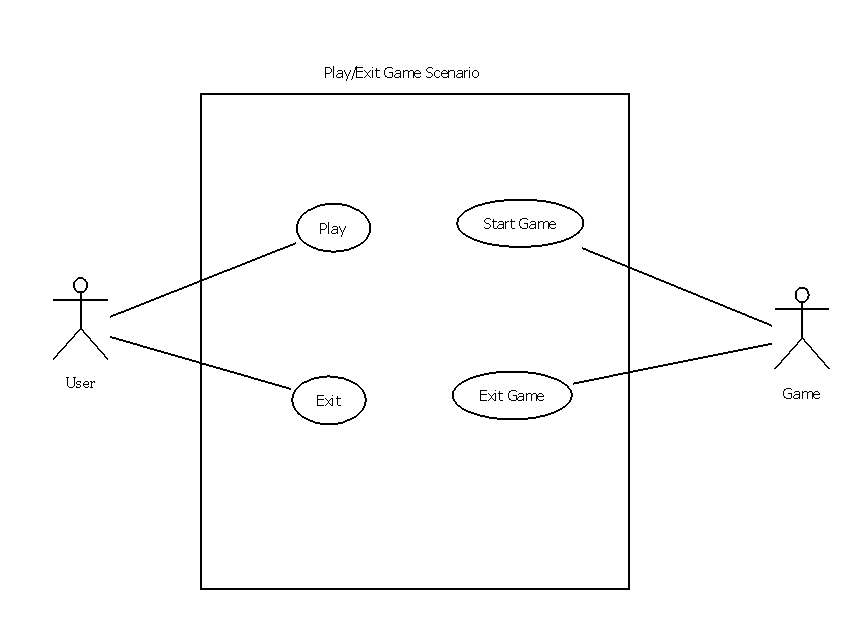
****

**4.3 Use Case Diagram**

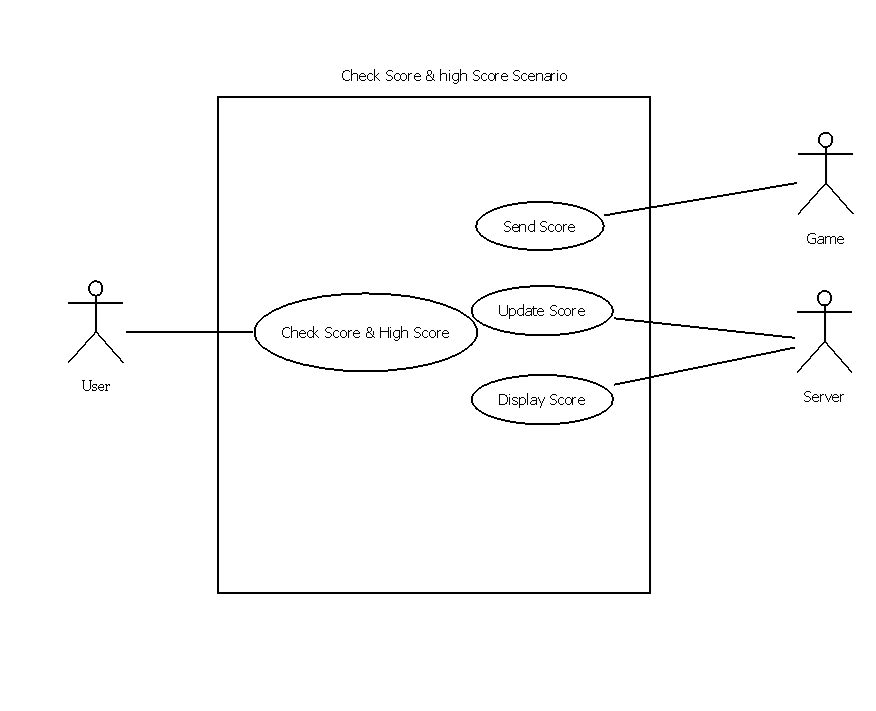
**Scenario1:**  **2048 Game**



**Scenario2: Play/Exit of 2048 Game**



**Scenario3: check score and highscore scenario of 2048 Game**



**4.4 Sequence Diagram**

**Scenario1: 2048 Game**



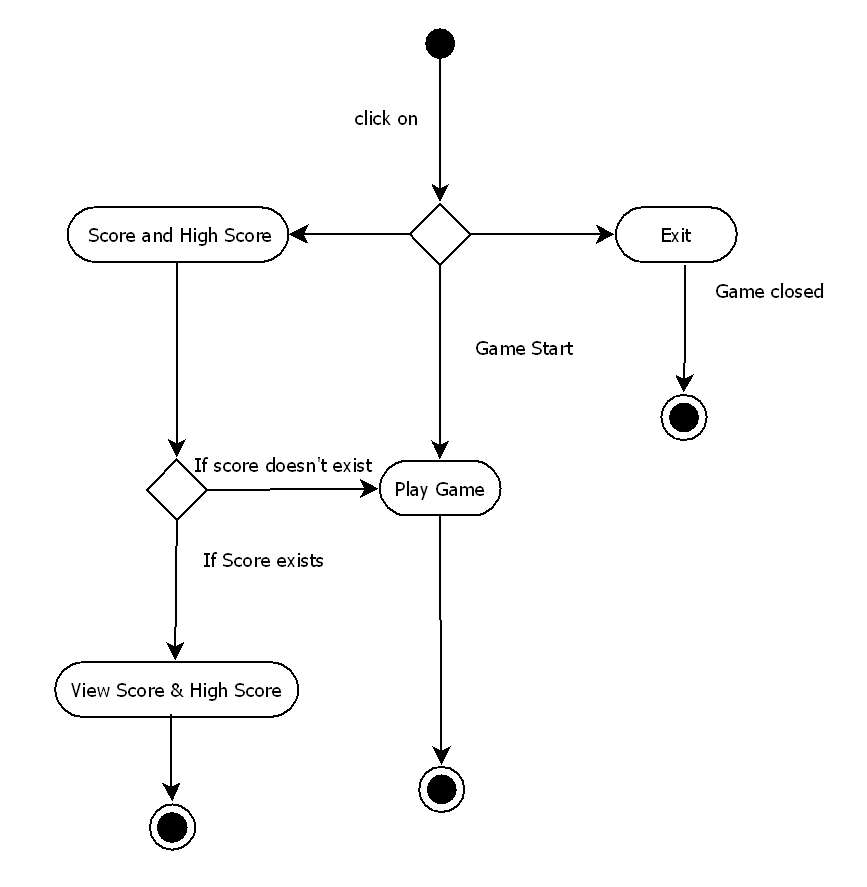
**Scenario2: Move Tiles**



**Scenario3: Calculate Score**

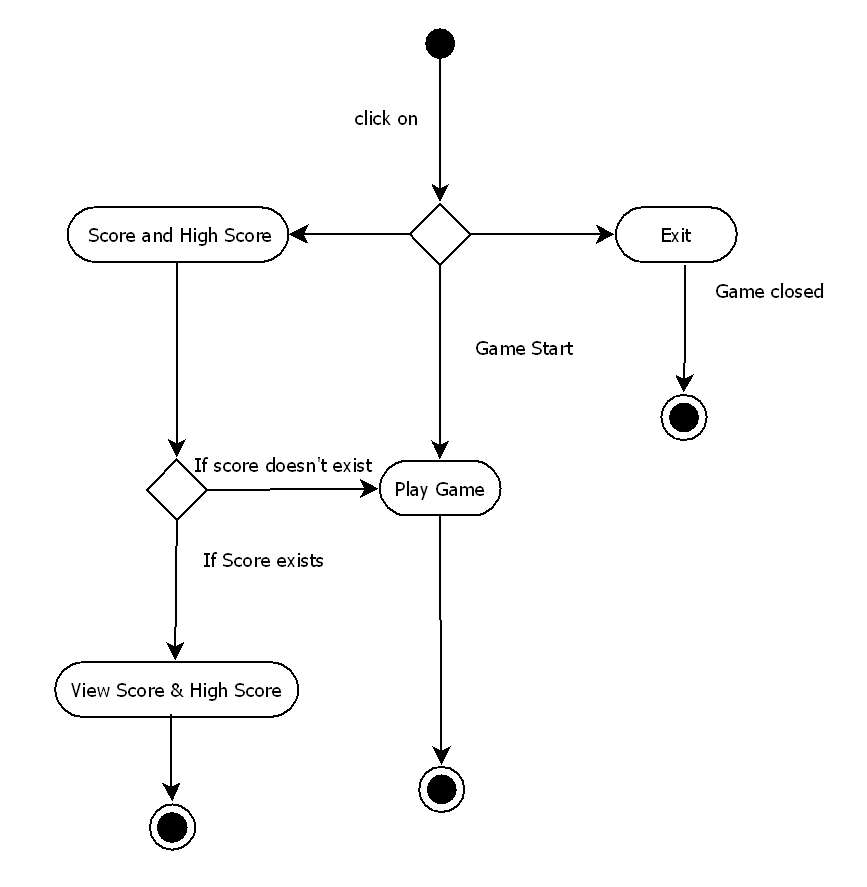


**4.5 Activity Diagram**

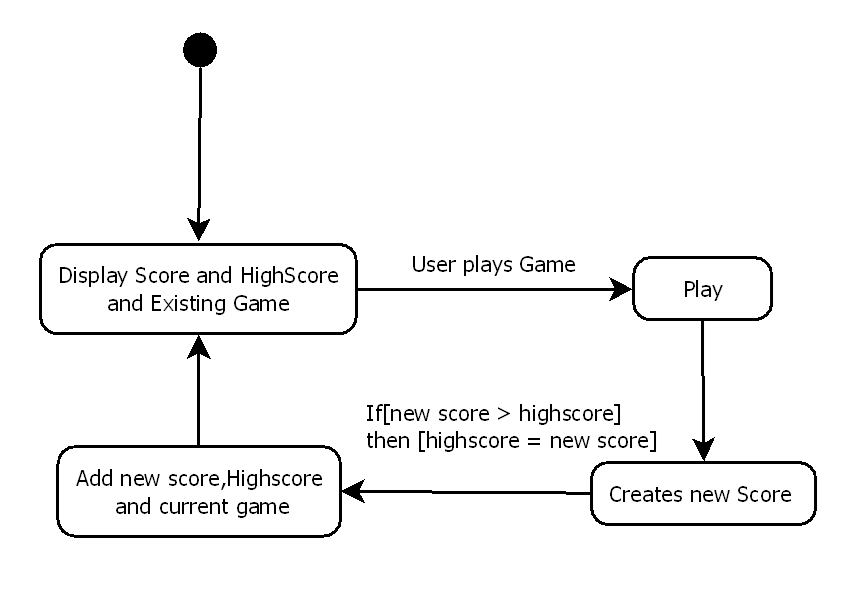


**4.6 State Diagram**

**Scenario1: User**



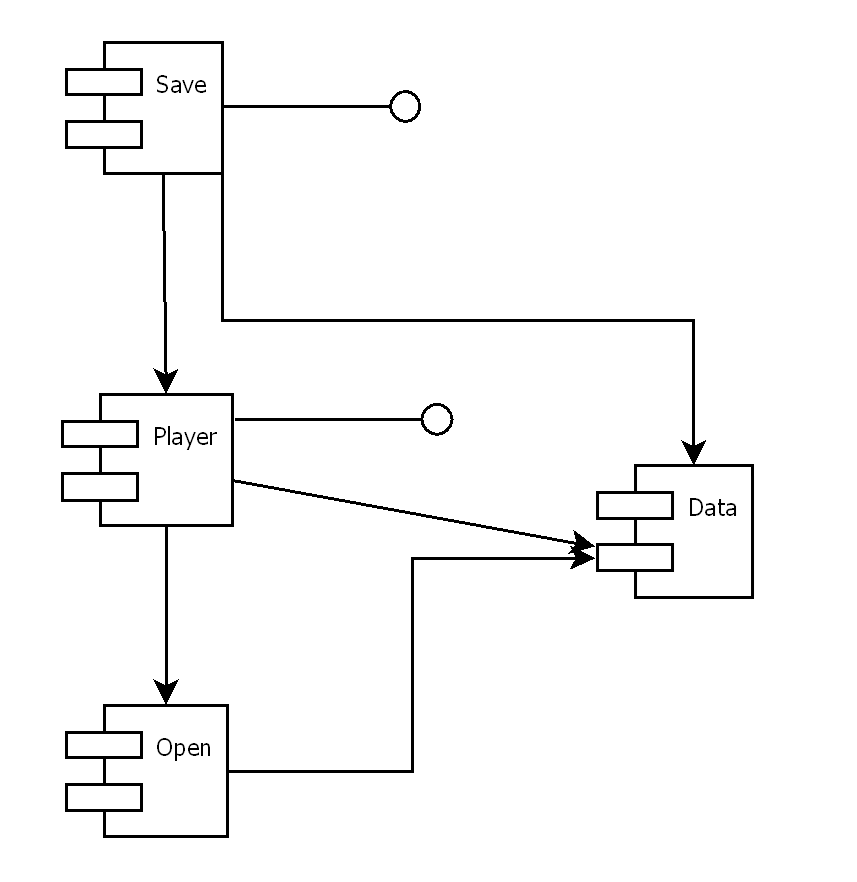
**Scenario2: Server**



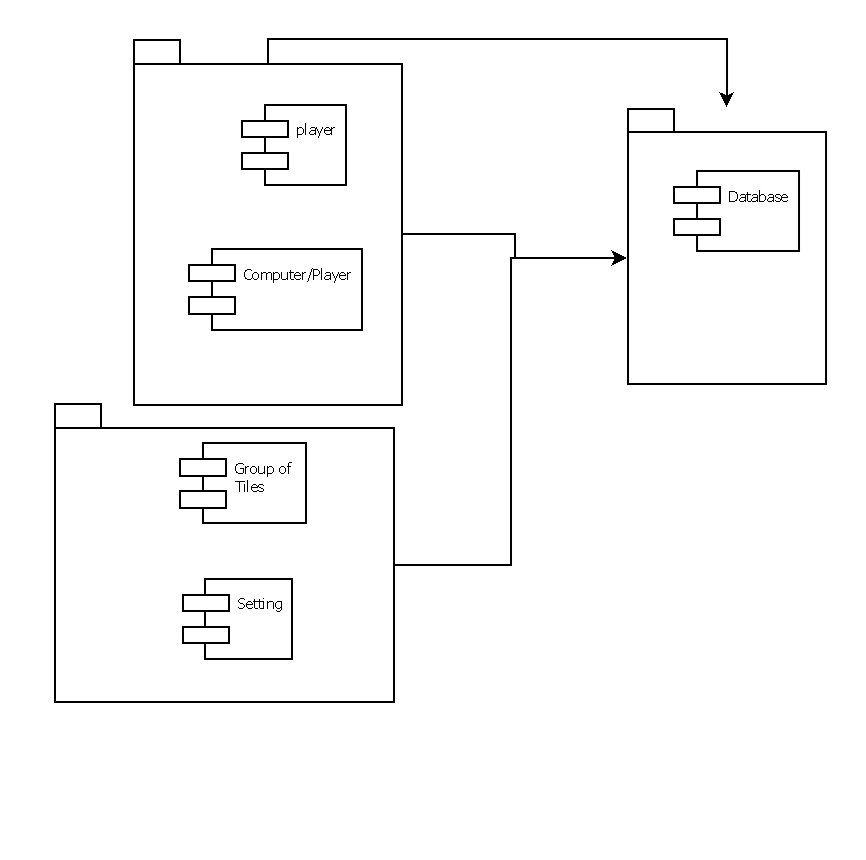
**4.7 Package Diagram**



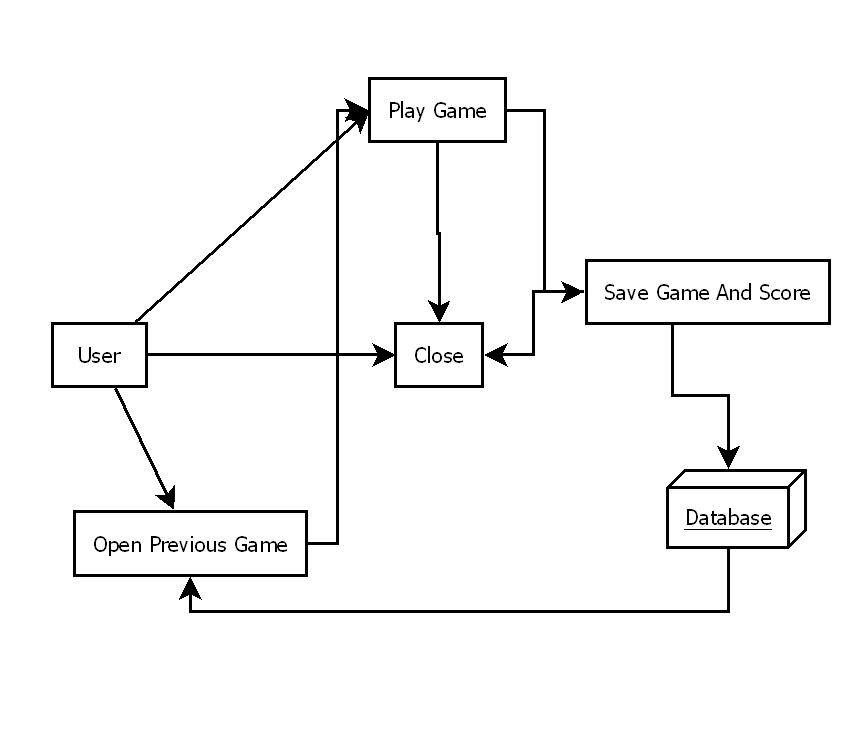
**4.8 Component Diagram**



**4.9 Deployment Diagram**

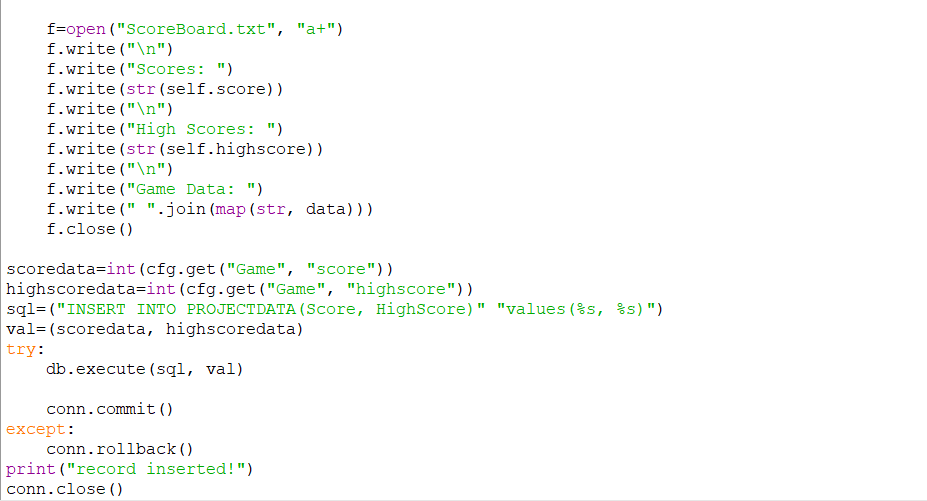


**4.10 Data Flow Diagram**



**4.11 Database Design**

****



**5. System Implementation**

**Game.py :**

# import standard

import sys

import time

import random

import configparser

import mysql.connector

#Throw connection using mysql-connector

conn =mysql.connector.connect(host="localhost",user="root",passwd="123456",database="Project")

db=conn.cursor()

db.execute("show tables")

lst=db.fetchall()

#checking if table exist or not and if not exist create one

table ="projectdata"

if (table,) in lst:

print("{} table alredy exists".format(table))

else:

print("{} table not exists".format(table))

db.execute("create table projectdata(Score varchar(20),HighScore varchar(20))")

conn.commit()

db.execute("show tables")

lst1 =db.fetchall()

print(lst1)

# import third-party

from PyQt5 import QtWidgets, QtGui, QtCore

# read configuration

cfg = configparser.ConfigParser()

cfg.read('settings.ini')

class Tile(QtCore.QObject):

def \_\_init\_\_(self, matrix, value):

super(Tile, self).\_\_init\_\_()

self.matrix = matrix

self.value = value

self.\_x = 0

self.\_y = 0

self.\_width = 0

self.\_height = 0

self.move\_animation = QtCore.QPropertyAnimation(self, b'geometry')

self.move\_animation.setDuration(int(cfg.get("Appearance", "time.animations")))

self.spawn\_animation = QtCore.QPropertyAnimation(self, b'geometry')

self.spawn\_animation.setDuration(int(cfg.get("Appearance", "time.animations")))

self.splash\_animation = QtCore.QPropertyAnimation(self, b'geometry')

self.splash\_animation.setDuration(int(cfg.get("Appearance", "time.animations")))

self.cell\_color = QtGui.QColor("#" + cfg.get("Appearance", "color.%s" % value if value <= 2048 else 2048))

self.text\_color = QtGui.QColor("#" + cfg.get("Appearance", "color.text.dark"))

if value > 4:

self.text\_color = QtGui.QColor("#" + cfg.get("Appearance", "color.text.light"))

self.font = QtGui.QFont()

def setGeometry(self, rect: QtCore.QRect):

self.\_x = rect.x()

self.\_y = rect.y()

self.\_width = rect.width()

self.\_height = rect.height()

self.matrix.parent.update()

def getGeometry(self):

if int(self.matrix.tl) == int(self.\_width):

return QtCore.QRect(self.\_x, self.\_y, self.\_width, self.\_height)

else:

return QtCore.QRect(int(self.\_x + (self.matrix.tl - self.\_width) / 2),

int(self.\_y + (self.matrix.tl - self.\_width) / 2),

self.\_width,

self.\_height)

def render(self, painter):

painter.setPen(self.cell\_color)

painter.setBrush(self.cell\_color)

painter.drawRoundedRect(self.getGeometry(),

self.matrix.sf \* 3,

self.matrix.sf \* 3,

QtCore.Qt.AbsoluteSize)

painter.setPen(self.text\_color)

pixel\_size = int(16 \* self.matrix.sf \* (self.\_width/self.matrix.tl)) if self.value < 1024 else \

int(15 \* self.matrix.sf \* (self.\_width / self.matrix.tl))

self.font.setPixelSize(pixel\_size if pixel\_size else 1)

painter.setFont(self.font)

painter.drawText(self.getGeometry(),

QtCore.Qt.AlignHCenter | QtCore.Qt.AlignVCenter,

str(self.value))

def spawn(self):

self.spawn\_animation.setStartValue(QtCore.QRect(self.\_x, self.\_y, int(self.matrix.tl/2), int(self.matrix.tl/2)))

self.spawn\_animation.setEndValue(QtCore.QRect(self.\_x, self.\_y, self.\_width, self.\_height))

self.spawn\_animation.start()

def splash(self):

self.splash\_animation.setStartValue(self.getGeometry())

self.splash\_animation.setKeyValueAt(0.5, QtCore.QRect(self.\_x,

self.\_y,

int(self.\_width + 7 \* self.matrix.sf),

int(self.\_height + 7 \* self.matrix.sf)))

self.splash\_animation.setEndValue(self.getGeometry())

self.splash\_animation.start()

def move(self, target: QtCore.QPoint):

self.move\_animation.setStartValue(self.getGeometry())

self.move\_animation.setEndValue(QtCore.QRect(target.x(), target.y(), self.\_width, self.\_height))

self.move\_animation.start()

# noinspection PyUnresolvedReferences

geometry = QtCore.pyqtProperty(QtCore.QRect, fset=setGeometry)

class Matrix:

def \_\_init\_\_(self, parent):

self.parent = parent # parent link for tiles updating

self.data = [] # tiles and coords massive

self.grid = int(cfg.get("Game", "grid")) # grid resolution

self.sf = 1 # current scale factor

self.tl = 37.5 # target tile length

self.sp = 4 # space beetwen tiles

self.modified = False # modified anchor

self.save\_loaded = False # loading progress anchor

save = cfg.get("Game", "save")

self.fill(list(map(int, save.split(" "))) if save else None)

def update(self):

self.tl = (150.0 / self.grid) \* self.sf # length of tile side

self.sp = (20.0 / (self.grid + 1)) \* self.sf # space beetwen tiles

for row in range(self.grid):

for cell in range(self.grid):

# update 'position' value for every cell on grid

x = int(20 \* self.sf + (cell + 1) \* self.sp + cell \* self.tl)

y = int(130 \* self.sf + (row + 1) \* self.sp + row \* self.tl)

self.data[row][cell]['position'] = QtCore.QPoint(x, y)

# update geometry for existing tiles

if self.data[row][cell]['data'][0]:

self.data[row][cell]['data'][0].setGeometry(QtCore.QRect(x, y, int(self.tl), int(self.tl)))

def fill(self, defaults=None):

# create empty

self.data = []

for row in range(self.grid):

self.data.append([])

for column in range(self.grid):

self.data[row].append({'position': None, 'data': []})

# create source

if defaults and len(defaults) == self.grid \*\* 2:

src = defaults

self.save\_loaded = True

else:

src = [0 for x in range(self.grid \*\* 2)]

# fill

counter = 0

for row in self.data:

for cell in row:

cell['data'].append(Tile(self, src[counter]) if src[counter] else 0)

counter += 1

def to\_render(self):

res = []

for row in self.data:

for cell in row:

if cell['data'][0]:

res.append(cell['data'][0])

if len(cell['data']) > 1:

res.append(cell['data'][1])

return res

def find\_empty\_cells(self):

res = []

for row in range(self.grid):

for cell in range(self.grid):

if self.data[row][cell]['data'][0] == 0:

res.append((row, cell))

return res

def spawn(self):

row, cell = random.choice(self.find\_empty\_cells())

tile = Tile(self, 4 if random.randrange(99) > 89 else 2)

tile.setGeometry(QtCore.QRect(self.data[row][cell]['position'].x(),

self.data[row][cell]['position'].y(),

int(self.tl), int(self.tl)))

self.data[row][cell]['data'] = [tile]

tile.spawn()

def collect(self):

for row in self.data:

for cell in row:

if len(cell['data']) > 1:

score = cell['data'][0].value + cell['data'][1].value

self.parent.score += score

if self.parent.score > self.parent.highscore:

self.parent.highscore = self.parent.score

new\_tile = Tile(self, score)

new\_tile.setGeometry(QtCore.QRect(cell['position'].x(),

cell['position'].y(),

int(self.tl), int(self.tl)))

cell['data'] = [new\_tile]

new\_tile.splash()

def shift\_tile\_left(self, r, c):

if c == 0 or len(self.data[r][c - 1]['data']) > 1:

return r, c

if self.data[r][c - 1]['data'][0] == 0:

self.data[r][c - 1]['data'] = self.data[r][c]['data']

self.data[r][c]['data'] = [0]

return self.shift\_tile\_left(r, c - 1)

if self.data[r][c - 1]['data'][0].value == self.data[r][c]['data'][0].value:

self.data[r][c - 1]['data'].append(self.data[r][c]['data'][0])

self.data[r][c]['data'] = [0]

return r, c - 1

return r, c

def merge(self):

start\_anim\_time = None

for r in range(self.grid):

for c in range(self.grid):

if self.data[r][c]['data'][0] != 0:

tile = self.data[r][c]['data'][0]

nr, nc = self.shift\_tile\_left(r, c)

if nc != c:

tile.move(self.data[nr][nc]['position'])

self.modified = True

start\_anim\_time = time.time()

if start\_anim\_time:

while time.time() - start\_anim\_time < int(cfg.get("Appearance", "time.animations")) / 1000:

self.parent.update()

def reverse(self):

data = []

for row in self.data:

new\_row = []

for cell in row:

new\_row.insert(0, cell)

data.append(new\_row)

self.data = data

def rotateLeft(self):

data = [[] for x in range(self.grid)]

for row in range(self.grid):

for cell in range(self.grid):

data[- 1 - cell].append(self.data[row][cell])

self.data = data

def rotateRight(self):

data = [[] for x in range(self.grid)]

for row in range(self.grid):

for cell in range(self.grid):

data[cell].insert(0, self.data[row][cell])

self.data = data

def backup(self):

res = []

for row in range(self.grid):

res.append([])

for cell in range(self.grid):

res[row].append({})

res[row][cell]['data'] = self.data[row][cell]['data'].copy()

return res

def check\_state(self):

# if empty cells exists check passed

if len(self.find\_empty\_cells()) > 0:

return True

# horizontal check cells for identity

for row in range(self.grid):

for cell in range(self.grid):

if cell:

if self.data[row][cell]['data'][0].value == self.data[row][cell - 1]['data'][0].value:

return True

# vertical check cells for identity

for row in range(self.grid):

for cell in range(self.grid):

if row:

if self.data[row][cell]['data'][0].value == self.data[row - 1][cell]['data'][0].value:

return True

# result if check not passed

return False

def find\_2048(self):

for row in self.data:

for cell in row:

if cell['data'][0] != 0:

if cell['data'][0].value == 2048:

return True

return False

class Canvas(QtWidgets.QWidget):

def \_\_init\_\_(self, parent=None):

super(Canvas, self).\_\_init\_\_(parent)

self.sf = 1

self.parent = parent

self.matrix = parent.matrix

self.painter = QtGui.QPainter()

self.new\_button = QtWidgets.QPushButton(cfg.get("Locale", "new"), self)

self.new\_button.setFocusPolicy(QtCore.Qt.NoFocus)

self.new\_button.clicked.connect(parent.new\_game)

self.undo\_button = QtWidgets.QPushButton(cfg.get("Locale", "undo"), self)

self.undo\_button.setFocusPolicy(QtCore.Qt.NoFocus)

self.undo\_button.clicked.connect(parent.undo)

with open('buttons.css', 'r') as css:

self.buttons\_style = css.read()

def paintEvent(self, event):

# open painter

self.painter.begin(self)

self.painter.setRenderHints(QtGui.QPainter.Antialiasing | QtGui.QPainter.TextAntialiasing)

# draw title

self.painter.setPen(QtGui.QColor("#" + cfg.get("Appearance", "color.text.dark")))

font = QtGui.QFont()

font.setPixelSize(int(self.sf \* 30))

self.painter.setFont(font)

self.painter.drawText(int(self.sf \* 20), int(self.sf \* 45), cfg.get("Locale", "subtitle"))

# draw help line

font.setPixelSize(int(self.sf \* 9))

self.painter.setFont(font)

text\_line\_rect = QtCore.QRect(int(self.sf \* 20), int(self.sf \* 65), int(self.sf \* 170), int(self.sf \* 20))

if self.parent.state == "win":

self.painter.drawText(text\_line\_rect, QtCore.Qt.AlignHCenter, cfg.get("Locale", "win"))

elif self.parent.state == "lose":

self.painter.drawText(text\_line\_rect, QtCore.Qt.AlignHCenter, cfg.get("Locale", "lose"))

else:

self.painter.drawText(text\_line\_rect, QtCore.Qt.AlignHCenter, cfg.get("Locale", "help"))

# draw score and best

st = " %s\n%s" % (cfg.get("Locale", "score"), self.parent.score)

hst = " %s\n%s" % (cfg.get("Locale", "best"), self.parent.highscore)

font.setPixelSize(int(self.sf \* 10))

self.painter.setFont(font)

sbr = self.painter.boundingRect(self.geometry(), QtCore.Qt.TextWordWrap, st)

hsbr = self.painter.boundingRect(self.geometry(), QtCore.Qt.TextWordWrap, hst)

sr = QtCore.QRect(int(self.sf \* 180) - sbr.width() - hsbr.width() - int((15 \* self.sf)),

int(self.sf \* 20),

sbr.width() + int(self.sf \* 10),

sbr.height() + int(self.sf \* 6))

hsr = QtCore.QRect(int(self.sf \* 180) - hsbr.width(),

int(self.sf \* 20),

hsbr.width() + int(self.sf \* 10),

hsbr.height() + int(self.sf \* 6))

self.painter.setPen(QtGui.QColor("#" + cfg.get("Appearance", "color.grid")))

self.painter.setBrush(QtGui.QColor("#" + cfg.get("Appearance", "color.grid")))

self.painter.drawRoundedRect(sr, int(self.sf \* 3), int(self.sf \* 3), QtCore.Qt.AbsoluteSize)

self.painter.drawRoundedRect(hsr, int(self.sf \* 3), int(self.sf \* 3), QtCore.Qt.AbsoluteSize)

self.painter.setPen(QtGui.QColor("#" + cfg.get("Appearance", "color.background")))

self.painter.drawText(sr, QtCore.Qt.AlignVCenter | QtCore.Qt.AlignHCenter, st)

self.painter.drawText(hsr, QtCore.Qt.AlignVCenter | QtCore.Qt.AlignHCenter, hst)

# draw boundary of the playing field

self.painter.setPen(QtGui.QColor("#" + cfg.get("Appearance", "color.grid")))

self.painter.setBrush(QtGui.QColor("#" + cfg.get("Appearance", "color.grid")))

playfield = QtCore.QRect(int(self.sf \* 20), int(self.sf \* 130), int(self.sf \* 170), int(self.sf \* 170))

self.painter.drawRoundedRect(playfield, int(self.sf \* 3), int(self.sf \* 3), QtCore.Qt.AbsoluteSize)

# draw grid of playing field

self.painter.setPen(QtGui.QColor("#" + cfg.get("Appearance", "color.cell")))

self.painter.setBrush(QtGui.QColor("#" + cfg.get("Appearance", "color.cell")))

ln, sp = 150.0 / self.matrix.grid, 20.0 / (self.matrix.grid + 1)

for y in range(self.matrix.grid):

for x in range(self.matrix.grid):

self.painter.drawRoundedRect(int(self.sf \* (20 + (x + 1) \* sp + x \* ln)),

int(self.sf \* (130 + (y + 1) \* sp + y \* ln)),

int(self.sf \* ln),

int(self.sf \* ln),

int(self.sf \* 3), int(self.sf \* 3),

QtCore.Qt.AbsoluteSize)

# draw all existing tiles

for tile in self.matrix.to\_render():

tile.render(self.painter)

# draw shadow if state is 'lose'

if self.parent.state == 'lose':

self.painter.setPen(QtGui.QColor(187, 173, 160, 100))

self.painter.setBrush(QtGui.QColor(187, 173, 160, 100))

self.painter.drawRoundedRect(playfield, int(self.sf \* 3), int(self.sf \* 3), QtCore.Qt.AbsoluteSize)

# close painter

self.painter.end()

def resizeEvent(self, ev):

sf = ev.size().width() / float(cfg.get("Appearance", "min.width")) # scale factor

self.sf = sf

self.matrix.sf = sf

self.matrix.update()

self.new\_button.setGeometry(int(20 \* sf), int(88 \* sf), int(75 \* sf), int(25 \* sf))

self.undo\_button.setGeometry(int(115 \* sf), int(88 \* sf), int(75 \* sf), int(25 \* sf))

dynamic\_style = self.buttons\_style % (cfg.get("Appearance", "color.grid"),

int(sf \* 3),

cfg.get("Appearance", "color.grid"),

int(sf \* 10),

cfg.get("Appearance", "color.background"))

self.new\_button.setStyleSheet(dynamic\_style)

self.undo\_button.setStyleSheet(dynamic\_style)

class Main(QtWidgets.QWidget):

def \_\_init\_\_(self):

super(Main, self).\_\_init\_\_()

self.setMinimumSize(int(cfg.get("Appearance", "min.width")), int(cfg.get("Appearance", "min.height")))

self.resize(int(cfg.get("Window", "width")), int(cfg.get("Window", "height")))

center\_point = QtWidgets.QDesktopWidget().availableGeometry().center()

qtrect = self.geometry()

qtrect.moveCenter(center\_point)

self.move(qtrect.topLeft())

self.setWindowTitle(cfg.get("Locale", "title"))

self.setAutoFillBackground(True)

pallete = self.palette()

pallete.setColor(self.backgroundRole(), QtGui.QColor("#" + cfg.get("Appearance", "color.background")))

self.setPalette(pallete)

self.state = "playing"

self.score = int(cfg.get("Game", "score"))

self.highscore = int(cfg.get("Game", "highscore"))

self.previous\_score = 0

self.previous\_matrix = None

self.matrix = Matrix(self)

self.canvas = Canvas(self)

self.show()

if not self.matrix.save\_loaded:

self.score = 0

self.matrix.spawn()

def keyPressEvent(self, event):

if self.state == "lose":

return

if not event.isAutoRepeat():

self.matrix.modified = False

self.previous\_matrix = self.matrix.backup()

self.previous\_score = self.score

if event.key() == QtCore.Qt.Key\_Left:

self.matrix.merge()

elif event.key() == QtCore.Qt.Key\_Up:

self.matrix.rotateLeft()

self.matrix.merge()

self.matrix.rotateRight()

elif event.key() == QtCore.Qt.Key\_Right:

self.matrix.reverse()

self.matrix.merge()

self.matrix.reverse()

elif event.key() == QtCore.Qt.Key\_Down:

self.matrix.rotateRight()

self.matrix.merge()

self.matrix.rotateLeft()

if self.matrix.modified:

self.matrix.collect()

self.matrix.spawn()

self.check\_state()

def new\_game(self):

self.previous\_score = self.score

self.previous\_matrix = self.matrix.backup()

self.score = 0

self.state = "playing"

self.matrix.fill()

self.matrix.update()

self.matrix.spawn()

def undo(self):

if self.previous\_matrix:

self.matrix.data = self.previous\_matrix

self.score = self.previous\_score

self.matrix.update()

self.check\_state()

self.previous\_matrix = None

def check\_state(self):

if self.matrix.check\_state() is False:

self.state = "lose"

elif self.matrix.find\_2048():

self.state = "win"

else:

self.state = "playing"

self.update()

def resizeEvent(self, event):

new\_size = event.size()

ar = float(cfg.get("Appearance", "aspect.ratio"))

if new\_size.width() != event.oldSize().width():

new\_width = new\_size.width()

new\_height = int(new\_width / ar)

if new\_height > new\_size.height():

new\_height = new\_size.height()

new\_width = int(new\_height \* ar)

else:

new\_height = new\_size.height()

new\_width = int(new\_height \* ar)

if new\_width > new\_size.width():

new\_width = new\_size.width()

new\_height = int(new\_width / ar)

self.canvas.resize(new\_width, new\_height)

self.canvas.move(int((self.width() - new\_width) / 2), int((self.height() - new\_height) / 2))

def closeEvent(self, event):

data = []

tiles\_count = 0

for row in self.matrix.data:

for cell in row:

if cell['data'][0] == 0:

data.append(0)

else:

data.append(cell['data'][0].value)

tiles\_count += 1

if tiles\_count == 1:

data = []

cfg.set("Game", "save", " ".join(map(str, data)))

cfg.set("Game", "score", str(self.score))

cfg.set("Game", "highscore", str(self.highscore))

cfg.set("Window", "width", str(self.canvas.width()))

cfg.set("Window", "height", str(self.canvas.height()))

with open('settings.ini', 'w') as target:

cfg.write(target)

event.accept()

f=open("ScoreBoard.txt", "a+")

f.write("\n")

f.write("Scores: ")

f.write(str(self.score))

f.write("\n")

f.write("High Scores: ")

f.write(str(self.highscore))

f.write("\n")

f.write("Game Data: ")

f.write(" ".join(map(str, data)))

f.close()

scoredata=int(cfg.get("Game", "score"))

highscoredata=int(cfg.get("Game", "highscore"))

print(scoredata)

sql=("INSERT INTO PROJECTDATA(Score, HighScore)" "values(%s, %s)")

val=(scoredata, highscoredata)

try:

db.execute(sql, val)

conn.commit()

except:

conn.rollback()

print("record inserted!")

conn.close()

if \_\_name\_\_ == "\_\_main\_\_":

app = QtWidgets.QApplication(sys.argv)

game = Main()

sys.exit(app.exec\_())

**Setting.ini**

[Game]

highscore = 0

grid = 4

save =

score = 0

[Window]

width = 650

height = 990

[Appearance]

min.width = 210

min.height = 320

aspect.ratio = 0.65625

time.animations = 150

color.background = faf8ef

color.text.dark = 776e65

color.text.light = f9f6f2

color.grid = bbada0

color.cell = cdc1b3

color.2 = eee4da

color.4 = ede0c8

color.8 = f2b179

color.16 = f59563

color.32 = f67c5f

color.64 = f65e3b

color.128 = edcf72

color.256 = edcc61

color.512 = edc850

color.1024 = edc53f

color.2048 = edc22e

[Locale]

title = 2048 Game

subtitle = 2048

score = Score

best = Best

help = Join the numbers and get to the 2048 tile!

win = CONGRATULATIONS!

lose = GAME OVER!

new = Restart

undo = Undo

**button.css**

QPushButton {

border: 1px solid #%s;

border-radius: %spx;

background-color: #%s;

font: %spx;

color: #%s;

}

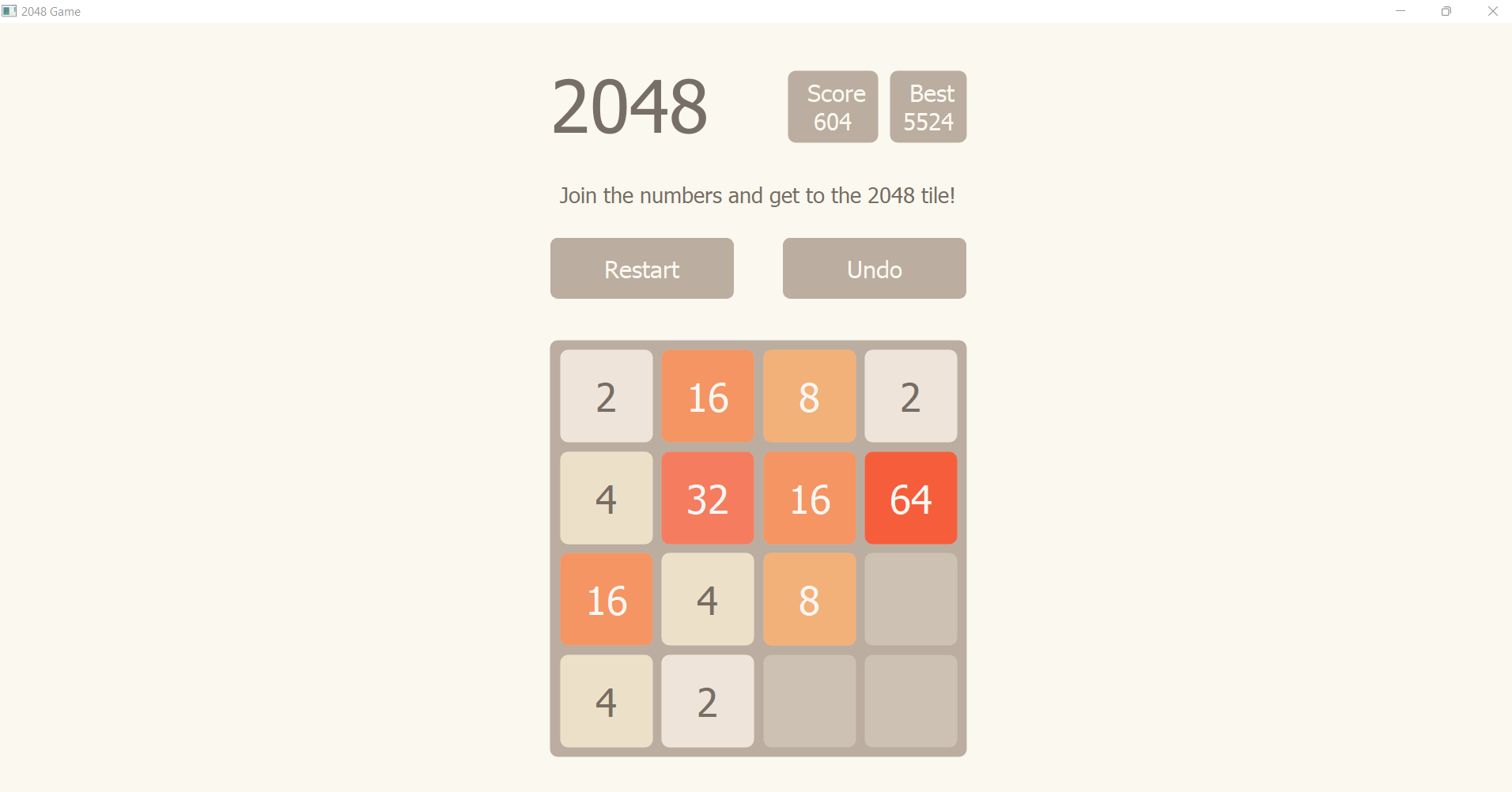
**6. Results**

**6.2 Validation**

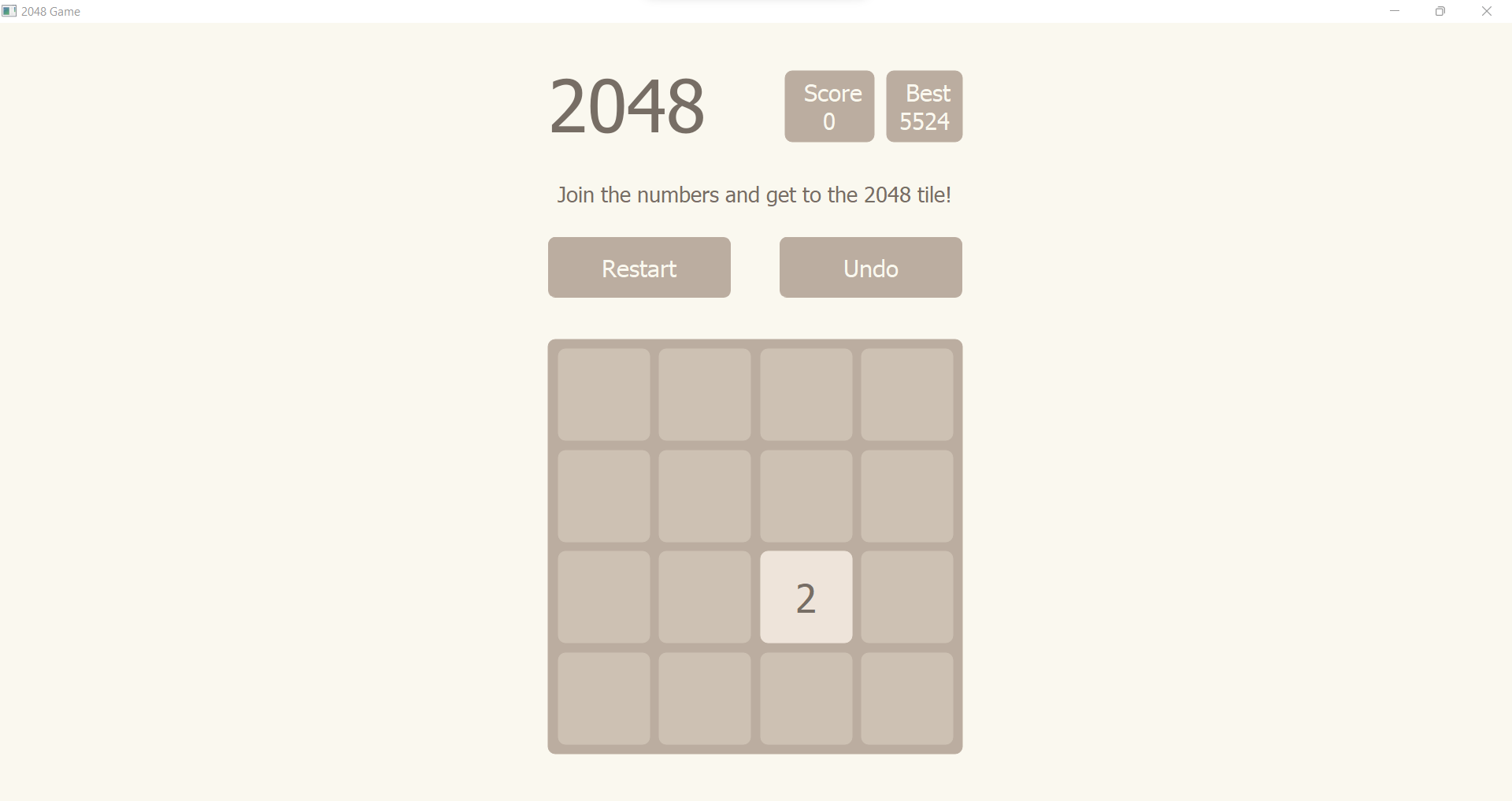
|  |  |
| --- | --- |
| RESTART | Restart the Game |
| Save | Save the current game |
| Open | Open pervious saved game |
| Quit | Close game |

**6.2 Screenshots**

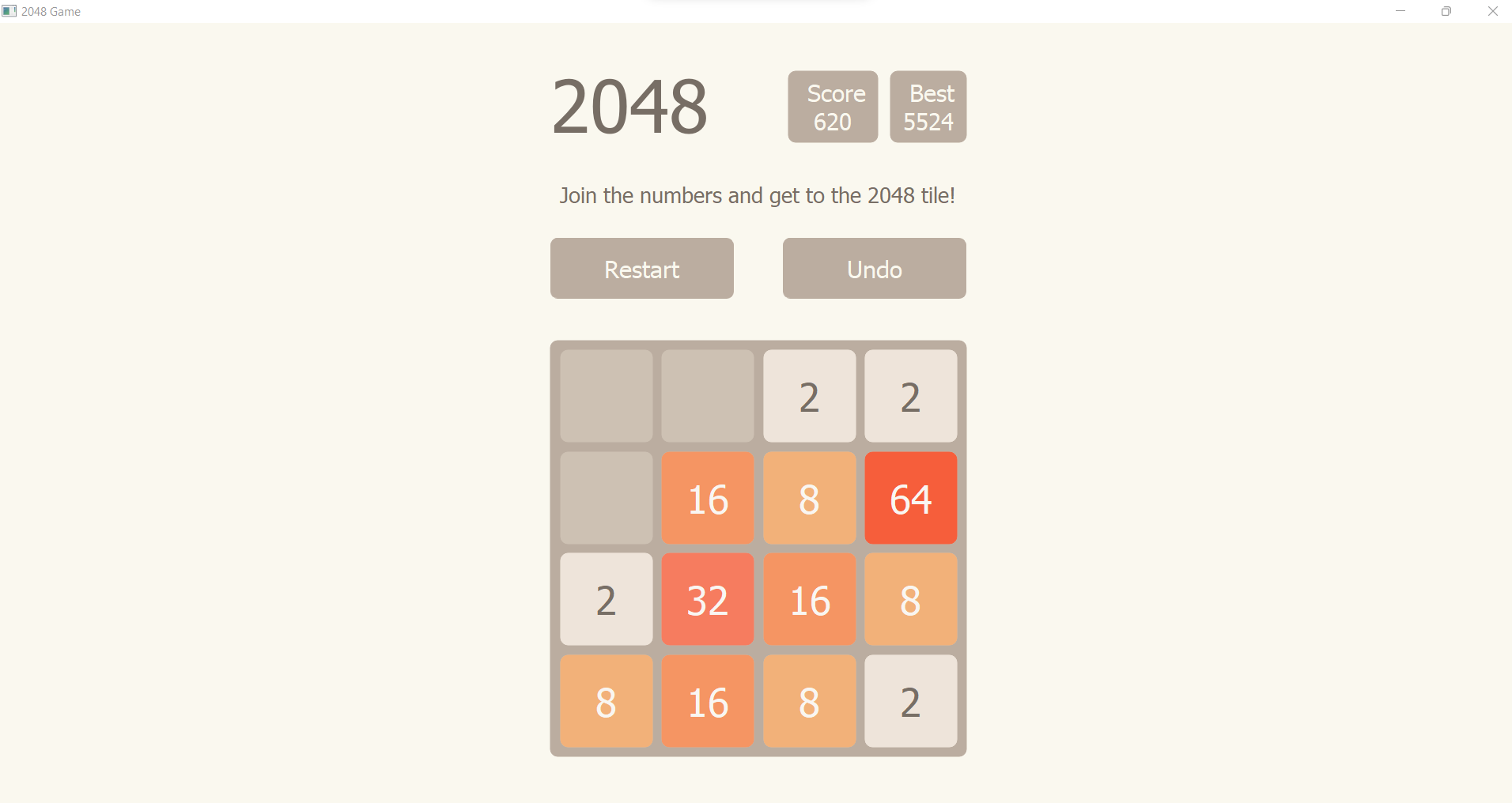
**6.2.1 Main Page**

****

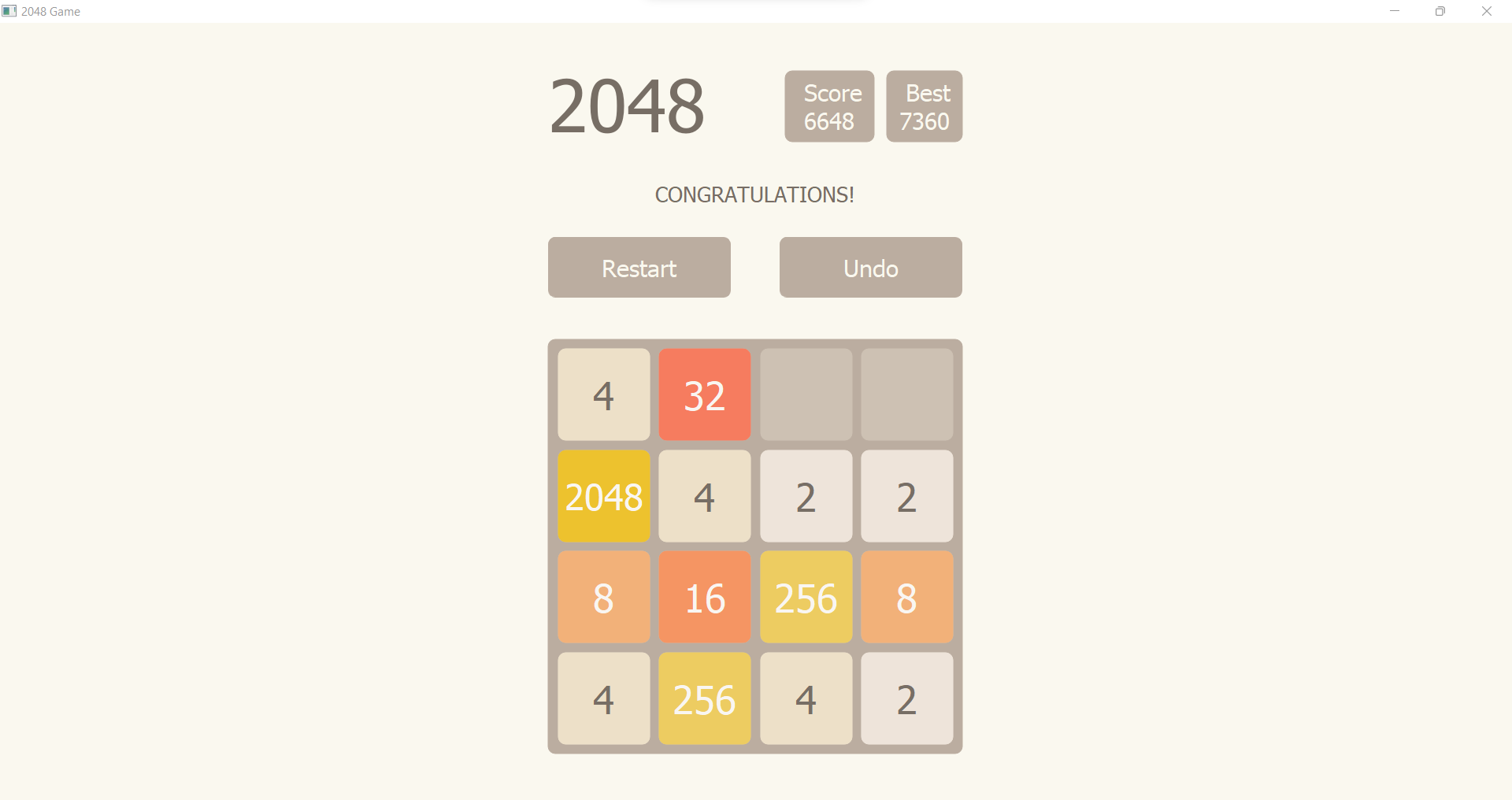
**6.2.2 Restart Option**

****

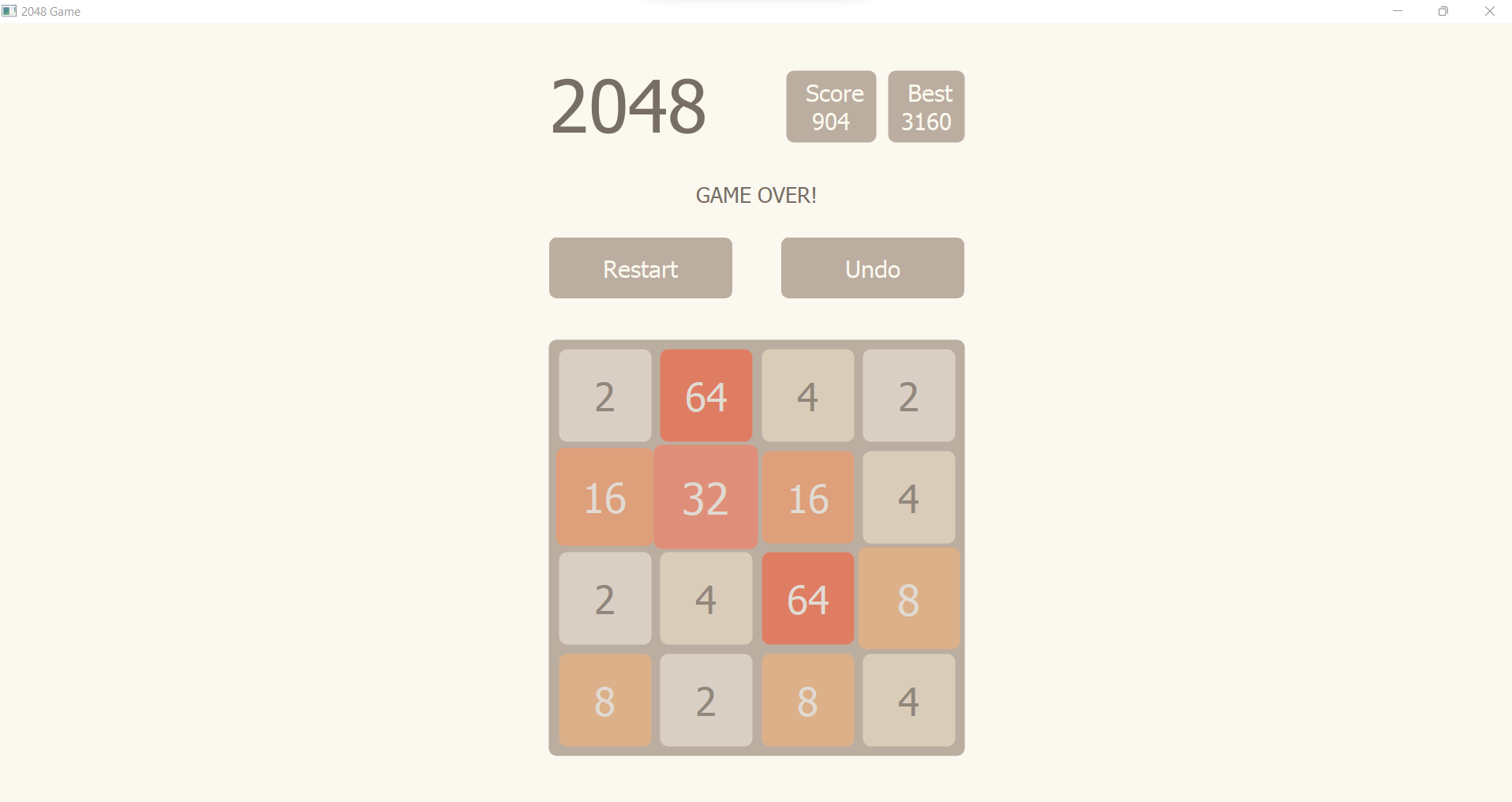
**6.2.3 Undo Option**



**6.2.4 Win**

****

**6.2.5 Lose**



**7. Conclusion and Future Scope**

**7.1 Future Enhancement**

* The system is modified in future as per the owner requirement.
* The 2048 will provide more features.
* There can be many more types that can be added.

**7.2 Conclusion**

* The system is error free and highly portable. It also has more option to future development.
* The system has a very good database handling and provide security.
* The approach of our system to maintain speed and accuracy.

**8. Reference**

* <https://www.w3schools.com/>
* <https://stackoverflow.com/>
* <https://www.youtube.com>
* https://www.github projects.com
* <https://www.google.com>

**9. Annexure**

**Figure List:-**

|  |  |  |
| --- | --- | --- |
| **Figure No** | **Name of Figure** | **Page No** |
| **4.2** | **Class Diagram** | **6** |
| **4.3** | **Use Case Diagram** | **7** |
| **4.4** | **Sequence Diagram** | **10** |
| **4.5** | **Activity Diagram** | **13** |
| **4.6** | **State Diagram** | **14** |
| **4.7** | **Package Diagram** | **16** |
| **4.8** | **Component Diagram** | **17** |
| **4.9** | **Deployment Diagram** | **18** |
| **4.10** | **Data Flow Diagram** | **19** |
| **6.1** | **Validation** | **43** |
| **6.2.1** | **Screenshot of Main Page** | **44** |
| **6.2.2** | **Screenshot of Restart** | **45** |
| **6.2.3** | **Screenshot of Undo** | **45** |
| **6.2.4** | **Screenshot of Win** | **46** |
| **6.2.5** | **Screenshot of Lose** | **46** |