UNIVERSITÀ
DI PARMA

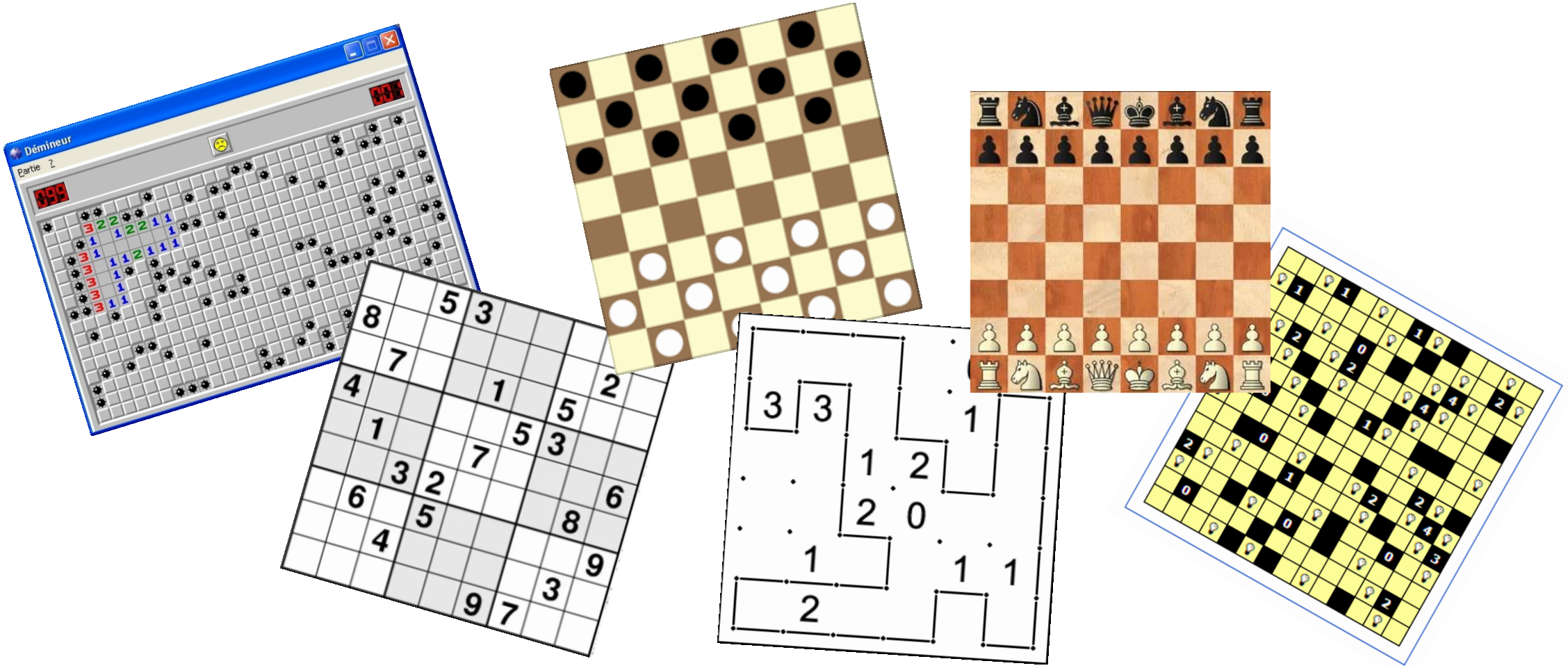
gioco da tavolo

board game

Alberto Ferrari

board game

- gioco da tavolo
- comprende una superficie di gioco suddivisa in caselle:
  - tabellone
  - scacchiera
  - ...
- sulla superficie sono presenti i pezzi del gioco che possono essere:
  - inseriti
  - eliminati
  - mossi da una casella a un'altra
  - cambiati di valore (immagine)
  - ...


Démineur
Partie 2

interfaccia astratta

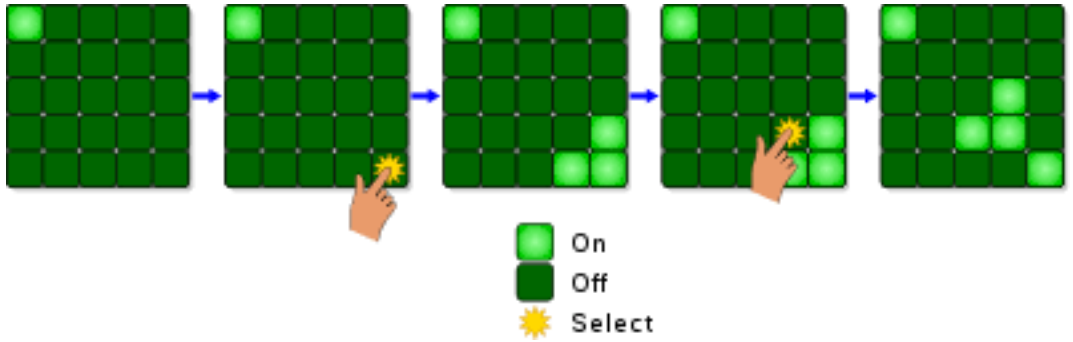
```
def abstract():
    raise NotImplementedError("Abstract method")

class BoardGame:
    def play_at(self, x: int, y: int): abstract()
    def flag_at(self, x: int, y: int): abstract()
    def value_at(self, x: int, y: int) -> str: abstract()
    def cols(self) -> int: abstract()
    def rows(self) -> int: abstract()
    def finished(self) -> bool: abstract()
    def message(self) -> str: abstract()
```

gioco a console

```
def print_game(game: BoardGame):
    for y in range(game.rows()):
        for x in range(game.cols()):
            print('{:3}'.format(game.value_at(x, y)), end='')
        print()

def console_play(game: BoardGame):
    print_game(game)

    while not game.finished():
        x, y = input().split()
        game.play_at(int(x), int(y))
        print_game(game)

    print(game.message())
```

Lights out!

- scacchiera m × n
- su ogni casella ci sono un interruttore e una lampadina
- ogni volta che si preme un interruttore, alcune lampadine cambiano di stato (se erano accese si spengono, e se erano spente si accendono) e sono:
  - la lampadina nella stessa casella dell'interruttore (*)
  - le lampadine nelle quattro caselle adiacenti (o meno di quattro, se la casella è sul bordo)



http://www.millstone.demon.co.uk/games/lightsout/lightsout.htm

Lights out! gioco a console


```
from boardgame import BoardGame, console_play
from random import randrange

class LightsOut(BoardGame):

    def __init__(self, side=5, level=4):
        self._cols, self._rows = side, side
        self._board = [[False for x in range(side)] for y in range(side)]
        for _ in range(level):
            self.play_at(randrange(side), randrange(side))

    def cols(self) -> int:
        return self._cols

    def rows(self) -> int:
        return self._rows

    def finished(self) -> bool:
        for y in range(self._rows):
            for x in range(self._cols):
                if self._board[y][x]:
                    return False
        return True
```

gioco a console ... play_at(x,y) ...

```
    def play_at(self, x: int, y: int):
        '''Place (or remove) a light at cell (x, y)'''
        if 0 <= x < self._cols and 0 <= y < self._rows:
            for dx, dy in ((0, 0), (0, -1), (1, 0),
                           (0, 1), (-1, 0)):
                x1, y1 = x + dx, y + dy
                if 0 <= x1 < self._cols and 0 <= y1 < self._rows:
                    self._board[y1][x1] = not self._board[y1][x1]

    def flag_at(self, x: int, y: int):
        pass

    def value_at(self, x: int, y: int) -> str:
        if (0 <= x < self._cols and 0 <= y < self._rows and self._board[y][x]):
            return '@'
        return '-'

    def message(self) -> str:
        return "Puzzle solved!"
```

```
def main():
    game = LightsOut()
    console_play(game)

main()
```

```
import g2d
from boardgame import BoardGame
from time import time

W, H = 40, 40
LONG_PRESS = 0.5

class BoardGameGui:
    def __init__(self, g: BoardGame):
        self._game = g
        self._downtime = 0
        self.update_buttons()

    def tick(self):
        if g2d.key_pressed("LeftButton"):
            self._downtime = time()
        elif g2d.key_released("LeftButton"):
            mouse = g2d.mouse_position()
            x, y = mouse[0] // W, mouse[1] // H
            if time() - self._downtime > LONG_PRESS:
                self._game.flag_at(x, y)
            else:
                self._game.play_at(x, y)
            self.update_buttons()
```

```
    def update_buttons(self):
        g2d.clear_canvas()
        g2d.set_color((0, 0, 0))
        cols, rows = self._game.cols(), self._game.rows()
        for y in range(1, rows):
            g2d.draw_line((0, y * H), (cols * W, y * H))
        for x in range(1, cols):
            g2d.draw_line((x * W, 0), (x * W, rows * H))
        for y in range(rows):
            for x in range(cols):
                value = self._game.value_at(x, y)
                center = x * W + W//2, y * H + H//2
                g2d.draw_text_centered(value, center, H//2)
        g2d.update_canvas()
        if self._game.finished():
            g2d.alert(self._game.message())
            g2d.close_canvas()
```

```
def gui_play(game: BoardGame):
    g2d.init_canvas((game.cols() * W, game.rows() * H))
    ui = BoardGameGui(game)
    g2d.main_loop(ui.tick)
```

Lights out! gioco gui


```
from boardgame import BoardGame
from boardgamegui import BoardGameGui, gui_
play
from random import randrange

…
```

```
game = LightsOut()
gui_play(game)
```