АННОТАЦИЯ

Данная работа включает в себя <кол-во рисунков> рисунков, <кол-во таблиц> таблиц, < кол-во листингов> листингов, <кол-во формул> формул и <кол-во приложений> приложений. Количество страниц в работе — <кол-во страниц>.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc184726810)

[1 ПОСТАНОВКА ЗАДАЧИ 5](#_Toc184726811)

[1 ЭТО ЗАГОЛОВОК 1-ГО УРОВНЯ (РАЗДЕЛ) 6](#_Toc184726812)

[1.1 Это заголовок 2-го уровня (ПОДРАЗДЕЛ) 6](#_Toc184726813)

[1.1.1 Это заголовок 3-го уровня (ПУНКТ) 6](#_Toc184726814)

[ЗАКЛЮЧЕНИЕ 10](#_Toc184726815)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 11](#_Toc184726816)

[ПРИЛОЖЕНИЯ 12](#_Toc184726817)

[Приложение <буква приложения>.<номер раздела при необходимости> 13](#_Toc184726818)

ВВЕДЕНИЕ

# 1 ПОСТАНОВКА ЗАДАЧИ

Цель работы: спроектировать и реализовать на базе ПЛИС подсистему ввода/вывода для устройства, предназначенного для игры в "Уголки" между двумя игроками. Подсистема должна обеспечивать взаимодействие пользователя с устройством посредством стандартных интерфейсов PS/2 (для клавиатуры) и VGA (для монитора).

* Провести анализ предметной области;
* Разработать системную модель подсистемы ввода/вывода;
* Разработать RTL-модель подсистемы ввода/вывода с использованием интерфейсов PS/2 и VGA;
* Сформулировать подходы к размещению подсистемы ввода/вывода;
* Создать макет пары устройств на базе ПЛИС, произведя интеграцию с логической подсистемой устройства;
* Провести верификацию;
* Составить отчетную документацию по проделанной работе.

11

# 2 СИСТЕМНАЯ МОДЕЛЬ УСТРОЙСТВА

## 2.1 Описание предметной области

Уголки — это настольная логическая игра для двух игроков, в которой игроки перемещают свои шашки по доске с целью занять дом противника. В данной работе рассматривается классический вариант игры.

В начале игры 12 шашек каждого игрока располагаются в домах – квадратах клетки в противоположных углах доски. Шашки первого игрока занимают левый нижний угол, шашки второго игрока – правый верхний.

Игроки ходят по очереди. За один ход игрок может передвинуть одну свою шашку.

Шашка может перемещаться на одну клетку по горизонтали или вертикали, если эта клетка свободна. Кроме того, шашка может "прыгать" через одну шашку (как свою, так и противника) по горизонтали или вертикали, если клетка за ней свободна. За один ход можно сделать несколько прыжков одной шашкой.

Цель игры - переместить все свои шашки в дом противника.

Игра заканчивается, когда один из игроков переместил все свои шашки в дом противника.

## 2.2 Реализация системной модели

Опишем код на Python с использованием библиотеки Tkinter, который представляет собой программную реализацию игры Уголки с графическим интерфейсом.

* Game: Основной класс, управляющий состоянием игры и взаимодействием с пользователем;
  + Board – двумерный массив 8x8, представляющий игровое поле;
  + current\_player – указывает на текущего игрока (1 или 2);
  + selected\_piece – хранит координаты выбранной шашки;
  + make\_move(x, y) – обрабатывает ход игрока, проверяет допустимость хода, обновляет состояние игрового поля;
  + is\_valid\_move(start, end) – проверяет, является ли ход допустимым согласно правилам игры;
  + move\_piece(start, end) – перемещает шашку на игровом поле;
  + check\_winner\_1() – проверяет, не достиг ли один из игроков условия победы;
  + check\_winner\_2() – определяется победитель по количеству шашек в чужом доме;
  + update\_buttons() – обновляет состояние кнопок в графическом интерфейсе в соответствии с состоянием board;
  + Button (из Tkinter) – представляет кнопки на игровом поле;
  + Command – связывает кнопку с функцией make\_move, вызываемой при нажатии.

Код системной модели представлен ниже (Приложение А).

## 2.3 Верификация системной модели



Рисунок <№ раздела>.<№ рисунка внутри раздела> — <Название рисунка с большой буквы>

Это ссылка на Таблицу <№ раздела>.<№ таблицы внутри раздела> из основного текста.

Таблица <№ раздела>.<№ таблицы внутри раздела> — <Название таблицы с большой буквы>

|  |  |
| --- | --- |
| Это заголовок столбца. Он центрируется по ширине столбца | Это заголовок столбца. Он центрируется по ширине столбца |
| Это содержимое таблицы | Это содержимое таблицы |
| Это содержимое таблицы | Это содержимое таблицы |
| Это содержимое таблицы | Это содержимое таблицы |
| Это содержимое таблицы | Это содержимое таблицы |

Продолжение Таблицы <№ раздела>.<№ таблицы внутри раздела>

|  |  |
| --- | --- |
| Это заголовок столбца. Он центрируется по ширине столбца | Это заголовок столбца. Он центрируется по ширине столбца |
| Это содержимое таблицы | Это содержимое таблицы |

Это первая строка после таблицы. Интервал перед: 6 пт.

(№ раздела.№ формулы)

Это текст после формулы.

ЗАКЛЮЧЕНИЕ

Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст. Это основной текст.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

ЭТО раздел литературы в списке источников

1. Это библиографическое описание источника.

2. Это библиографическое описание источника.

# ПРИЛОЖЕНИЯ

Приложение А — Системная модель на языке Python

Приложение <буква приложения> — <Название приложения с большой буквы>

Приложение <буква приложения> — <Название приложения с большой буквы>

### Приложение А

Системная модель на языке Python

Листинг 2.1 — Системная модель на языке Python

|  |
| --- |
| import tkinter as tk  from tkinter import messagebox  class Game:  def \_\_init\_\_(self, master):  self.master = master  self.master.title("Уголки")  self.master.geometry("536x542")  self.board\_size = 8  self.current\_player = 1  self.move\_count = 0  self.board = [[0] \* self.board\_size for \_ in range(self.board\_size)]  self.selected\_piece = None  self.create\_widgets()  self.reset\_board()  def create\_widgets(self):  self.buttons = [[None] \* self.board\_size for \_ in range(self.board\_size)]  for i in range(self.board\_size):  for j in range(self.board\_size):  btn = tk.Button(self.master, text='', font=('Arial', 24), width=3,  command=lambda x=i, y=j: self.make\_move(x, y))  btn.grid(row=i, column=j)  self.buttons[i][j] = btn  self.reset\_button = tk.Button(self.master, text='Сброс', command=self.reset\_board)  self.reset\_button.grid(row=self.board\_size, column=0, columnspan=self.board\_size)  def reset\_board(self):  self.board = [[0] \* self.board\_size for \_ in range(self.board\_size)]  for i in range(self.board\_size):  for j in range(self.board\_size):  self.buttons[i][j].config(text='', state=tk.NORMAL)  if j > 3 and i < 3:  self.board[i][j] = 1  self.buttons[i][j].config(text='X')  elif j < 4 and i > 4:  self.board[i][j] = 2  self.buttons[i][j].config(text='O')  self.update\_buttons()  self.current\_player = 1  self.selected\_piece = None  self.move\_count = 0  def update\_buttons(self):  for i in range(self.board\_size):  for j in range(self.board\_size):  if self.board[i][j] == 1:  self.buttons[i][j].config(text='X', state=tk.NORMAL)  elif self.board[i][j] == 2:  self.buttons[i][j].config(text='O', state=tk.NORMAL)  else:  self.buttons[i][j].config(text='', state=tk.NORMAL)  def make\_move(self, x, y):  if self.selected\_piece:  if self.is\_valid\_move(self.selected\_piece, (x, y)):  self.move\_piece(self.selected\_piece, (x, y))  self.move\_count += 1  if self.check\_winner():  messagebox.showinfo("Игра окончена", f"Игрок {self.current\_player} выиграл!")  self.reset\_board()  elif self.move\_count >= 80:  messagebox.showinfo("Игра окончена", "Ничья!")  self.reset\_board()  else:  self.current\_player = 2 if self.current\_player == 1 else 1  self.selected\_piece = None  else:  self.selected\_piece = None  self.buttons[x][y].config(relief=tk.RAISED) # Reset button relief  else:  if self.board[x][y] == self.current\_player:  self.selected\_piece = (x, y)  self.buttons[x][y].config(relief=tk.SUNKEN)  def is\_valid\_move(self, start, end):  sx, sy = start  ex, ey = end  if not (0 <= sx < self.board\_size and 0 <= sy < self.board\_size and  0 <= ex < self.board\_size and 0 <= ey < self.board\_size):  return False  if self.board[ex][ey] != 0:  return False  if abs(sx - ex) == 2 and sy == ey:  mid = (sx + ex) // 2  if self.board[mid][sy] != 0:  return True  if abs(sy - ey) == 2 and sx == ex:  mid = (sy + ey) // 2  if self.board[sx][mid] != 0:  return True  return False  def move\_piece(self, start, end):  sx, sy = start  ex, ey = end  self.board[ex][ey] = self.current\_player  self.board[sx][sy] = 0  self.update\_buttons()  def check\_winner(self):  player1\_in\_player2\_home = 0  player2\_in\_player1\_home = 0  for i in range(self.board\_size):  for j in range(self.board\_size):  if j > 3 and i < 3 and self.board[i][j] == 2:  player2\_in\_player1\_home += 1  elif j < 4 and i > 4 and self.board[i][j] == 1:  player1\_in\_player2\_home += 1  return player1\_in\_player2\_home == 12 or player2\_in\_player1\_home == 12  if \_\_name\_\_ == "\_\_main\_\_":  root = tk.Tk()  game = Game(root)  root.mainloop() |