



국민대학교
소프트웨어융합대학
소프트웨어학부


소프트웨어 프로젝트2

프로젝트

프로젝트 명	틱택토 게임 만들기
팀 명	H조
문서 제목	AD 프로젝트_최종 보고서

Version	1.2
Date	2021-11-27

팀원	이채영
	임민석
	이다현

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조


CONFIDENTIALITY/SECURITY WARNING

이 문서에 포함되어 있는 정보는 국민대학교 소프트웨어융합대학 소프트웨어학부 및 소프트웨어학부 개설 교과목 소프트웨어 프로젝트2 수강 학생 중 프로젝트 "틱택토 게임 만들기"를 수행하는 팀 "H조"의 팀원들의 자산입니다. 국민대학교 소프트웨어학부 및 팀 "H조"의 팀원들의 서면 허락없이 사용되거나, 재가공 될 수 없습니다.

문서 정보 / 수정 내역

Filename	최종보고서-틱택토 게임 구현.doc
원안작성자	이채영, 이다현, 임민석
수정작업자	이채영, 이다현, 임민석

수정날짜	대표수정자	Revision	추가/수정 항목	내 용
2021-11-30	이채영	1.0	최초 수정	보고서 목차 수정
2021-12-03	임민석	1.1	내용 추가	2.2.2와 2.2.3 내용 추가
2021-12-03	이다현	1.2	내용 추가	2.2.1, 2.2.4, 2.2.5 내용 추가
2021-12-05	임민석	1.2.1	내용 수정	개요 내의 pyinstaller 패키지 사용법 수정 및 참고문헌 수정
2021-12-05	임민석	1.2.2	내용 수정	2.2.2에서 언급된 각각의 함수들마다 개발자가 누구인지 명시

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

목 차

1	개요	4
2	개발 내용 및 결과물	6
2.1	목표	6
2.2	개발 내용 및 결과물	8
2.2.1	개발 내용	8
2.2.2	시스템 구조 및 설계도	10
2.2.3	활용/개발된 기술	15
2.2.4	현실적 제한 요소 및 그 해결 방안	17
2.2.5	결과물 목록	18
3	자기평가	18
4	참고 문헌	19
5	부록	20
5.1	사용자 매뉴얼	오류! 책갈피가 정의되어 있지 않습니다.
5.2	설치 방법	오류! 책갈피가 정의되어 있지 않습니다.

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

1 개요

우리 조는 사람수와 게임판의 크기를 설정할 수 있는 틱택토 게임을 개발하는 것을 목표로 정했다. 게임 시작창과 게임창 두 개로 나누어 두 코드를 연결하여 사용하는 것으로 하고, 초기화가 가능하도록 정했다. '혼자하기'를 누를 경우, 우리가 만들어 낸 'AI'코드를 활용해 승리를 위한 가장 최적의 위치에 말을 놓도록 설정하였고, '둘이하기'를 누를 경우, 누를 때마다 말을 바꾸어 두 명이서 게임을 할 수 있도록 구성했다. 또한 기존의 틱택토 방식인 3x3 뿐만 아니라 4x4와 5x5까지 구현하여 사용자들이 보다 다양하게 게임을 즐길 수 있도록 계획을 짰다.

처음엔 가장 기초가 되는 둘이서 하는 3x3버전의 틱택토를 세 명이 각자 코드를 짜오고, 그 중 가장 확장성이 높은 것으로 골라 가지를 뺐어나갔다. 둘이 하기는 비교적 쉽게 모두 만들어냈지만 혼자하기를 만들어내는 과정에서 어려움을 겪었다. 단순히 무작위로 컴퓨터가 말을 놓는 것이 아니라, 이길 수 있는 곳에 놓도록 구현하는 과정에서 생각해야 할 것들이 많았기 때문이다. 몇 번의 실패와 시행착오 끝에 최종적인 틱택토 게임을 구현해냈다.

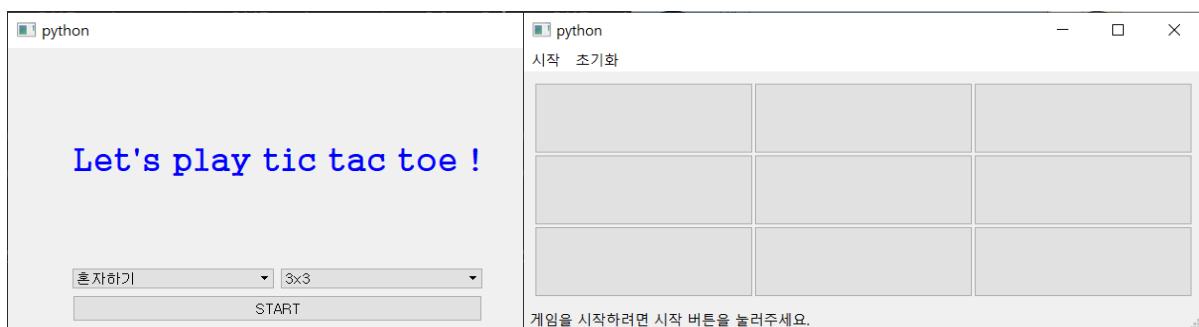


그림 1. 틱택토 게임 시작창과 게임창

구현을 끝낸 코드를 파이썬 패키지인 'pyinstaller'를 사용해 파이썬 파일을 실행파일(.exe)로 바꾸어 사용자가 쉽게 사용할 수 있도록 하였다. 과정은 다음과 같다.(단, 윈도우를 기준으로 한다.)

- ① 아나콘다에서 CMD.exe Prompt를 실행.

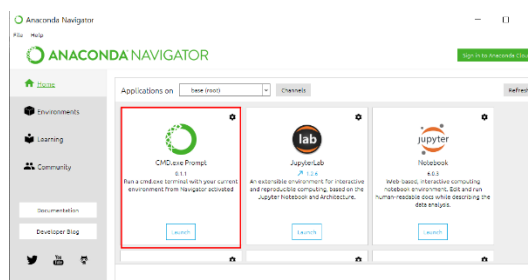



그림 2. 아나콘다

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

- ② 아나콘다 CMD창에서 'pip install pyinstaller'라고 입력한 후 엔터




그림 3. 아나콘다 CMD창

- ③ CMD창에서 'cd 경로(위치)' 명령어를 통해 자신이 .exe로 바꿀 .py가 있는 폴더의 경로로 이동
- ④ 'pyinstaller 파이썬 파일이름.py' 명령어를 입력하여 원하는 파일을 exe로 변환
- ⑤ 변환을 성공하면, exe로 변환하려고 했던 .py 파일 디렉토리 안 "dist" 디렉토리에 .exe 파일과 .py에 import 된 라이브러리가 들어있게 된다.



그림 4. 전환을 성공한 화면

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

2 개발 내용 및 결과물

2.1 목표

적용단계	내용	적용 여부
1단계	자료수집	적용
2단계	아키텍처 설계	적용
3단계	소프트웨어 설계	적용
4단계	소프트웨어 구현	적용
5단계	소프트웨어 시험	적용

1단계: 자료수집

우리 조는 '틱택토'게임을 gui로 구현하기로 결정하였고 거기서 더 나아가 인원과 게임 판의 크기를 조절할 수 있는 기능을 추가하기로 하였다. 먼저 틱택토 게임이란 무엇인지 조사하여 게임의 규칙과 구조를 알아내고, 쉽게 찾을 수 있는 소스코드들을 살펴보며 어떻게 짜는 것이 효율적일지 생각했다.


2단계: 아키텍처 설계

클래스를 어떻게 나누어야 효율적일지 고민하였다. 특히 코드에서 불필요한 반복은 코드의 길이만 늘릴 뿐 효율적이지 못하고 쉽게 이해하는데 방해가 된다고 생각했다. 그렇기 때문에 반복문으로 구현하는 것을 잊지 않고 구조적으로 큰 복잡함이 없도록 코드 구성을 진행하기로 하였다. 또한 클래스와 메소드 간의 관계를 생각하며 구조를 구성하기로 했다.

3단계: 소프트웨어 설계

수업시간에 다룬 PyQt5를 사용하고, 클래스는 게임창과 시작창을 구현하는 클래스, 총 2개로 나누어서 설계하였다. 오류가 나지 않도록 하는 것을 가장 큰 목표로 두고, 사용자가 사용할 때 불편함이 없도록 하는 것을 두번째 목표로 두었다. 게임을 사용하는 사람의 수에 따라서 진행되는 코드가 완전히 다르기 때문에 그 부분을 염두해놓고 설계했다.


4단계: 소프트웨어 구현

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

앞서 계획한 것들에 맞춰서 코드를 구현하였고 그 과정에서 가장 큰 목표는 '예상치 못한 오류를 생각해내어 해결하는 것'이었다. 앞서 ADS와 DDS를 작성하여(별첨 참조) 상세하게 계획을 짜 놓았기 때문에 그 계획에 맞춰 구현하도록 했다. 특히 구현하는 과정에서 AI코드('혼자하기'에서 사용될 코드)가 생각해내기 어렵고 복잡했기 때문에 더욱 중점을 두고 구현하였다.

5단계; 소프트웨어 시험

Unittest를 사용해서 게임을 진행하는데 오류가 발생하는지 확인하도록 했다. 만약 발생한다면 어느 부분이 문제인지 찾아보고 해결할 수 있도록 한다. 또한 우리가 직접 사용자가 되어서 써보고 혹시 모르는 부분에 대한 오류를 찾아내도록 한다.

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

2.2 개발 내용 및 결과물


2.2.1 개발 내용

1. 자료수집 - 틱택토 게임을 구현하고 이해를 위해 검색과 자료들을 수집해 게임의 규칙과 구조를 파악하고 어떻게 코드를 구성할 것인지 생각했다.
2. 설계 - 자료를 수집하고 설계를 생각하던 중에 반복으로 인한 코드의 길이가 길었고 우리는 반복을 줄이고 코드의 길이를 줄이고 복잡하지 않게 구현하기로 생각했다. 틱택토라는 게임의 코드들이 상당히 많았기에 우리는 조금 다르게 혼자 할 수 있는 AI를 추가하고 인원과 게임판의 크기를 조정할 수 있는 것을 설계하기로 했다. PyQt5를 사용하며 시작창과 게임창의 클래스를 나누어 만들기로 했다.
3. 게임의 시작창 구현 - 시작창을 만들고 시작창에 필요한 버튼(시작하기), 멘트, 정렬버튼(혼자하기, 둘이하기)(3x3, 4x4, 5x5)을 QLabel, QComboBox, QPushButton등을 이용해 위치에 맞게 구현했다.



그림 5. 틱택토 시작창

4. 게임창 구현

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

4-1. 3x3을 먼저 구현했다. 게임창에 필요한 시작, 초기화 메뉴바, 게임판, 승패를 가릴 수 있는 함수, 표식을 저장하는 함수등을 사용해 3x3을 먼저 구현했다.

4-2. 3x3과 같은 원리로 4x4, 5x5까지 구현했다.

4-3. 따로 만들었던 3x3, 4x4, 5x5를 합치는 과정에서 불필요하고 반복되는 코드를 줄여 간결하게 만들었고, 게임창을 구현했다.




그림 6. 틱택토 게임창

4-4. 혼자 플레이할 때 필요한 AI를 구현했다. 혼자 게임을 실행할 때 AI가 필요한 기능들을 생각하고 구현해보았고 AI가 본인이 이길 수 있는지 확인하고 그 자리에 놓는 알고리즘과 상대가 이긴다면 막는 알고리즘을 구현하여 추가했다. 혼자하기까지 구현을 완성했다.

5. 시작창과 게임창을 연결 - QStackedWidget을 사용해 선택을 하고 시작을 누르면 시작창에서 게임창으로 화면이 전환되게 했다.

6. 만든 파일들을 PyInstaller를 사용하여 실행파일로 만들었다.

7. 소프트웨어 시험 - 게임을 하는데 문제나 오류가 없는지 unittest를 사용하여 확인했다.

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

2.2.2 시스템 구조 및 설계도

시스템을 설계하는 과정에서 2 가지 소스 파일을 이용했다. 하나는 StartWindow.py, 다른 하나는 GameUI.py 이다.

1. StartWindow.py

StartWindow.py 는 메인 화면 구성과 화면 전환 역할을 맡고 있다. 이 소스 파일은 StartWindow 라는 클래스 안에 메인 화면 인터페이스를 구성하는 `__init__` 함수와 화면 전환 기능을 위해 스택형 위젯의 인덱스 값을 지정하는 `setWindow` 함수로 이루어져 있다.

1.1 메인 – 개발자: 이채영, 임민석

우선 메인을 살펴보면, 화면 전환을 위해서 스택형 위젯을 만들고, 그 안에 메인 화면 인스턴스와 게임 화면 인스턴스들을 순서대로 넣는다. 스택형 위젯은 위젯에 넣은 인스턴스 순서대로 인덱스 값을 지정하며, 이 인덱스의 값에 따라 화면이 달라질 수 있도록 도와준다. 이 때 게임 화면 인스턴스를 선언할 때 사용한 Game 메서드는 GameUI.py 파일과 연결되는 것이며, 형태는 Game(게임판의 크기, AI 사용 여부)이다.

```
if __name__ == '__main__':
    app = QApplication(sys.argv)


    # 화면 전환용 widget 생성
    widget = QtWidgets.QStackedWidget()
    for i in range(7):
        if i == 0:
            window = StartWindow()
        elif i < 4:
            window = Game(i+2, False)
        # i = 4, 5, 6
        else:
            window = Game(i-1, True)
        widget.addWidget(window)

    widget.show()
```

그림 7. StartWindow.py 의 메인

1.2 `__init__` 함수 – 개발자: 이채영, 이다현

`__init__` 함수는 메인 화면에 "Let's play Tic Tac Toe !" 문구를 띄우고, 플레이 할 게임 설정을 위해 콤보 박스 2 개와 Start 버튼을 생성한다. 그리고 Start 버튼을 눌렀을 시 `setWindow` 함수와 연결이 되도록 하고, `setWindow` 함수에서 쓸 변수 `self.initialSetting` 을 사전 형태로 선언한다.

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

```

def __init__(self):
    super().__init__()
    self.initialSetting = [{'3x3': 1, '4x4': 2, '5x5': 3}, {'3x3': 4, '4x4': 5, '5x5': 6}]

    start_sentence1 = QLabel("Let's play tic tac toe !", self)
    start_sentence1.setFont(QFont("궁서", 20))
    start_sentence1.setStyleSheet("Color : blue")
    self.howMany = QComboBox(self)
    self.howMany.addItem("혼자하기")
    self.howMany.addItem("둘이하기")

    self.nxn = QComboBox(self)
    self.nxn.addItem("3x3")
    self.nxn.addItem("4x4")
    self.nxn.addItem("5x5")

    self.startButton = QPushButton("START")
    self.startButton.clicked.connect(self.setWindow)

```

그림 8. 클래스 StartWindow 에서의 __init_ 함수 중 메인 화면 인터페이스 구성 코드

1.3 setWindow 함수 – 개발자: 이채영, 임민석

setWindow 함수는 Start 버튼이 눌렀을 때 콤보 박스로 설정한 값에 맞는 게임 화면으로 전환할 수 있도록 스택형 위젯의 인덱스 값을 조정한다. 만약 '혼자하기'를 골랐다면 게임판의 크기에 따라 인덱스 값이 4, 5, 6 중 하나가 된다. '둘이하기'를 골랐다면 게임판의 크기에 따라 인덱스 값이 1, 2, 3 중 하나가 된다.

```

def setWindow(self):
    if self.howMany.currentText() == '혼자하기':
        mainSettingdic = self.initialSetting[1]
    else :
        mainSettingdic = self.initialSetting[0]
    widget.setCurrentIndex(mainSettingdic[self.nxn.currentText()])

```

그림 9. setWindow 함수

2. GameUI.py

GameUI.py 는 게임 화면 구성 역할을 맡고 있다. 이 소스 파일은 Game 이라는 클래스 안에 게임 화면 인터페이스 구성과 초기 설정 값들을 선언하는 __init__ 함수가 있고, 게임 시작을 알리는 start 함수, 게임을 초기화하는 reset 함수, 각각의 게임 버튼마다 눌렀을 시 ai 사용 여부와 함께 chk 함수가 실행되도록 연결하는 SetButtonConnected 함수, 누른 버튼에 따라 그 버튼의 위치에 알맞은 모양을 넣고 조건에 따라 AI 함수도 실행하는 chk 함수, '혼자하기'일 때 상대 턴을 대신하여 수행해주는 AI 함수, 게임의 승패를 가리는 Checking 함수, 마지막으로 AI 지능 알고리즘인 AICheckingMark 함수와 AIDefenseMark 함수가 있다.

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

2.1 __init__ 함수 - 개발자: 이채영, 이다현, 임민석

먼저 __init__ 함수를 보자면, 받아오는 게임판의 크기 값과 모양의 초기값을 선언한다. 그리고 메뉴바에서 시작 버튼과 초기화 버튼을 누를 수 있도록 생성한다.

```
def __init__(self, num, ai):
    super().__init__()
    self.index = num
    self.mark = "0"
    self.setWindowTitle("틱택토")

    self.menu = self.menuBar()
    self.menu.addAction("시작", self.start)
    self.menu.addAction("초기화", self.reset)

    self.statusBar().showMessage("게임을 시작하려면 시작 버튼을 눌러주세요.")
```

그림 10. GameUI.py 의 클래스 Game 에서의 __init__ 함수

그 다음, 받아들이는 게임판의 크기 값에 따라 게임판을 구성한다. 이때 self.arr 배열은 화면에 직접적으로 나타나는 게임판 배열이고, self.win 배열은 게임의 승패 여부를 가리기 위해 사용되는 독립적인 배열이다. 또한 self.sw 는 게임의 시작 여부를 판별하는 변수이다.

```
self.arr = []
self.win = []
self.sw = 0 #게임의 시작상태 저장, 비활성화 라는 뜻


for r in range(self.index):
    self.arr.append([])
    self.win.append([])
    for c in range(self.index):
        self.arr[r].append(QPushButton(""))
        self.win[r].append("")
        self.arr[r][c].setSizePolicy(QSizePolicy.Expanding, QSizePolicy.Expanding)
        self.arr[r][c].setObjectName(str(r)+str(c))
        self.arr[r][c].setFont(self.fnt)
        self.grid.addWidget(self.arr[r][c], r, c)
```

그림 11. __init__ 함수의 받아들이는 게임판의 크기 값에 따라 게임판을 구성하는 코드

마지막으로 각각의 버튼마다 클릭했을 시 ai 의 사용 여부와 함께 chk 함수와 연결이 되도록 SetButtonConnected 함수를 실행하면 된다.

```
self.SetButtonConnected(self.index, ai)
```

그림 12. __init__ 함수의 SetButtonConnected 함수를 실행하는 코드

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

```
def SetButtonConnected(self, num, ai):
    if num == 3:
        self.arr[0][0].clicked.connect(lambda: self.chk(0, 0, ai))
        self.arr[0][1].clicked.connect(lambda: self.chk(0, 1, ai))
        self.arr[0][2].clicked.connect(lambda: self.chk(0, 2, ai))
        self.arr[1][0].clicked.connect(lambda: self.chk(1, 0, ai))
        self.arr[1][1].clicked.connect(lambda: self.chk(1, 1, ai))
        self.arr[1][2].clicked.connect(lambda: self.chk(1, 2, ai))
        self.arr[2][0].clicked.connect(lambda: self.chk(2, 0, ai))
        self.arr[2][1].clicked.connect(lambda: self.chk(2, 1, ai))
        self.arr[2][2].clicked.connect(lambda: self.chk(2, 2, ai))
```

그림 13. SetButtonConnected 함수의 내부 중 일부

2.2 start 함수와 reset 함수 – 개발자: 이채영, 이다현, 임민석

start 함수와 reset 함수는 메뉴 버튼인 '시작' 버튼과 '초기화' 버튼을 누르면 실행되는 함수이다. start 함수는 self.sw 변수를 1로 선언함으로써 게임의 시작을 알리고, reset 함수를 통해 게임판을 새로 초기화한다. reset 함수는 self.arr 배열에 있는 버튼들을 공백으로 초기화하고, 모양도 초기값으로 바꾸는 기능을 수행한다.


```
def start(self):
    self.sw = 1
    self.reset()

def reset(self):
    if self.sw == 1:
        for r in range(self.index):
            for c in range(self.index):
                self.arr[r][c].setText("")
        self.mark = "0"
```

그림 8. start 함수와 reset 함수

2.3 chk 함수 – 개발자: 이채영, 이다현, 임민석

chk 함수는 행과 열, AI의 사용 여부를 받아들이는 값을 토대로 클릭한 위치의 버튼이 턴수에 맞는 모양으로 바뀌게 만드는 역할을 맡고 있다.

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

```
def chk(self, row, column, ai):
    if self.sw == 1:
        if self.arr[row][column].text() != "":
            return
        else:
            self.arr[row][column].setText(self.mark)
            for r in range(self.index):
                for c in range(self.index):
                    self.win[r][c] = self.arr[r][c].text()
```

그림 9. chk 함수의 게임에서 클릭한 버튼이 턴 수에 맞는 모양으로 바뀌게 하는 코드

또한 게임의 승패 여부를 판단하여 게임을 비활성화 시키는 기능과 AI 의 사용 여부에 따라 AI 함수를 실행할 것인지에 대한 트리거를 담당하기도 한다.

```
# 게임판의 크기에 맞추어 게임의 승패 여부 가리는 함수 실행
if self.Checking():
    self.sw = 0


if self.mark == "0":
    self.mark = "X"
else:
    self.mark = "0"

# ai가 True일 때 실행
if ai:
    self.AI()
```

그림 10. chk 함수의 게임의 승패 여부를 판단하는 것과 매 턴마다 모양이 바뀔 수 있도록 self.mark 변수 조정, AI 사용 여부에 따른 AI 함수 트리거 기능을 구현한 코드

2.4 Checking 함수 – 개발자: 이채영, 임민석

Checking 함수는 게임이 이겼는지 졌는지, 아니면 비겼는지 판단하는 역할을 맡는다. 우선 승패 여부를 가리기 위해 self.win 배열의 가로, 세로, 대각선을 확인해서 만약 승리 조건에 부합하면 어떤 모양이 승리했는지 표시하고 True 를 반환한다.

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

```
def Checking(self):
    for i in range(self.index):
        rawsum = self.win[i][0]
        colsum = self.win[0][i]
        updiag = self.win[0][0]
        downdiag = self.win[0][self.index - 1]
        for j in range(1, self.index):
            rawsum += self.win[i][j]
            colsum += self.win[j][i]
            updiag += self.win[j][j]
            downdiag += self.win[j][self.index - (1 + j)]

    if self.mark * self.index == rawsum or self.mark * self.index == colsum or \
       self.mark * self.index == updiag or self.mark * self.index == downdiag:
        QMessageBox.about(self, "종료", self.mark + "가 이겼습니다!")
    return True
```

그림 11. Checking 함수의 게임 승패 여부를 가리는 기능을 구현한 코드

만약 승패가 가려지지 않았다면 count 변수를 통해 게임판 배열인 self.arr 에 모양이 있는 버튼의 수만큼 count 수가 증가하도록 하며, 만약 count 값이 게임판의 크기와 동일하다면 비긴 것으로 표시하고 True 를 반환한다.

```
count = 0
for r in range(self.index):
    for c in range(self.index):
        if self.arr[r][c].text() != '':
            count += 1

if count == self.index * self.index:
    QMessageBox.about(self, "종료", "비겼습니다!")
    return True

return False
```

그림 12. Checking 함수의 승패가 가려지지 않았을 시 비겼는지 판단하는 코드


2.5 AI 함수, AICheckingMark 함수, AIDefenseMark 함수 – 개발자: 이채영, 임민석

AI 함수, AICheckingMark 함수, AIDefenseMark 함수는 2.2.3 에서 상세하게 서술함.

2.2.3 활용/개발된 기술

1. AI 함수

이 프로젝트를 수행하면서 AI 알고리즘을 새로 고안하였다. AI 함수가 실행되면 self.arr 배열을 모방할 array 배열과 self.arr 배열 중 비어있는 배열의 인덱스 값을 저장하기 위한 emptyIndex 배열, 상황에 맞는 행동을 AI 가 할 수 있도록 도와주는 self.breakSwitch 변수를 선언한다.

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

```
def AI(self):
    if self.sw == 1:
        array = [[" " for cal in range(self.index)] for raw in range(self.index)]
        emptyIndex = []
        self.breakSwitch = 0
```

그림 13. AI 함수의 실행됐을 시 변수들 초기화하는 코드

현재 게임판의 상태를 array 배열에 그대로 복사하고 만약 비어 있는 버튼의 위치가 있다면 그 위치를 저장하기 위해 emptyIndex 배열에 [행, 열] 형태로 추가한다.

```
for r in range(self.index):
    for c in range(self.index):
        array[r][c] = self.arr[r][c].text()
        if self.arr[r][c].text() == " ":
            emptyIndex.append([r, c])
```

그림 14. AI 함수의 현재 게임판의 상태를 array 배열에 저장하고, emptyIndex 배열에 비어 있는 버튼의 위치를 추가하는 코드

emptyIndex 배열에 있는 비어 있는 버튼의 위치를 하나씩 꺼내어 지능 알고리즘인 AICheckingMark 함수와 AIDefenseMark 함수를 실행한다. 만약 이 두 함수 중 하나가 작동 된다면 self.breakSwitch 는 1 이 되므로 반복문을 벗어나고, 두 함수 모두 작동 되지 않고 self.breakSwitch 가 0 으로 유지 된다면 비어 있는 버튼의 위치 중 하나를 랜덤으로 골라 선택하는 행동을 취한다.

```
for index in emptyIndex:
    a = index[0]
    b = index[1]

    self.AICheckingMark(array, a, b)
    self.AIDefenseMark(array, a, b)


    if self.breakSwitch == 1:
        break

    if self.breakSwitch == 0:
        i = random.randrange(0, len(emptyIndex))
        self.chk(emptyIndex[i][0], emptyIndex[i][1], False)
```

그림 15. AI 함수의 지능 알고리즘 부분

2. AICheckingMark 함수와 AIDefenseMark 함수

AICheckingMark 함수와 AIDefenseMark 함수는 각각 AI 차례에서 AI 가 이길 수 있는지 판단하는 알고리즘, AI 차례에서 막지 않는다면 상대가 이길 수 있는지 판단하는

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

알고리즘이다. Checking 함수를 응용하여 array 배열의 가로, 세로, 대각선을 확인한 뒤 만약 받아들이는 비어 있는 버튼의 위치가 AI가 이기게 만드는 위치라면 그 위치의 버튼을 누르는 행동을, 상대가 다음 턴에 이길 수 있는 위치라면 그 위치의 버튼을 누름으로써 막는 행동을 하도록 구현했다.

```
def AICheckingMark(self, array, a, b):
    array[a][b] = self.mark
    for i in range(self.index):
        rawsum = array[a][b] if a == i and b == 0 else array[i][0]
        colsum = array[a][b] if a == 0 and b == i else array[0][i]
        updiag = array[a][b] if a == 0 and b == 0 else array[0][0]
        downdiag = array[a][b] if a == 0 and b == self.index - 1 else array[0][self.index - 1]
        for j in range(1, self.index):
            rawsum += array[a][b] if a == i and b == j else array[i][j]
            colsum += array[a][b] if a == j and b == i else array[j][i]
            updiag += array[a][b] if a == j and b == j else array[j][j]
            downdiag += array[a][b] if a == j and b == self.index - (1 + j) else array[j][self.index - (1 + j)]

    if self.mark * self.index == rawsum or self.mark * self.index == colsum or \
       self.mark * self.index == updiag or self.mark * self.index == downdiag:
        self.chk(a, b, False)
        self.breakSwitch = 1
        break
```

그림 16. AICheckingMark 함수


```
def AIDefenseMark(self, array, a, b):
    for i in range(self.index):
        rawsum = "0" if a == i and b == 0 else array[i][0]
        colsum = "0" if a == 0 and b == i else array[0][i]
        updiag = "0" if a == 0 and b == 0 else array[0][0]
        downdiag = "0" if a == 0 and b == self.index - 1 else array[0][self.index - 1]
        for j in range(1, self.index):
            rawsum += "0" if a == i and b == j else array[i][j]
            colsum += "0" if a == j and b == i else array[j][i]
            updiag += "0" if a == j and b == j else array[j][j]
            downdiag += "0" if a == j and b == self.index - (1 + j) else array[j][self.index - (1 + j)]

    if "0" * self.index == rawsum or "0" * self.index == colsum or \
       "0" * self.index == updiag or "0" * self.index == downdiag:
        self.chk(a, b, False)
        self.breakSwitch = 1
        break
```

그림 17. AIDefenseMark 함수

2.2.4 현실적 제한 요소 및 그 해결 방안

1. 시작창에서 플레이창으로 넘어가는 구현 방법을 어떻게 해야하는지 어려움을 겪었고 검색을 통해 QStackedWidget 을 사용하여 시작창에서 플레이창으로 화면을 전환하는

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

Widget 을 사용하여 문제를 해결했다.

2. AI 알고리즘 구현에서 많은 생각이 필요했고 AI 가 본인이 이길 수 있는지 확인하고 그 자리에 놓을 수 있는 알고리즘을 넣었지만 AI 가 쉽게 지는 것 같은 느낌이 들었고 상대가 이길 것 같을 때 막을 수 있는 함수를 구현하여 AI 의 기능을 더 추가했다.


2.2.5 결과물 목록

1. StartWindow.exe - 사용자가 쉽게 사용할 수 있도록 Pyinstaller를 이용해 실행파일로 바꾼 파일.
2. StartWindow.py - 게임을 시작하는 시작창. 각각의 버튼을 이용하여 실행 스타일을 선택할 수 있다.
3. GameUI.py - 게임창을 구현하는 함수들과 승패를 가리거나 저장하는 함수 AI함수가 입력되어 있다.

3 자기평가

이채영: '틱택토' 게임에 대한 아이디어를 제공하였고 거기서 어떠한 방향으로 나아가 게임을 확장할지 의견을 제시하였다. 게임 시작 화면의 기초코드를 작성했고, 다른 팀원이 짠 코드를 반복문을 사용해 줄이는 작업을 하였다. 주석을 활용해서 팀원들이 코드를 이해하는데 어려움이 없도록 했어야 하는데 그러지 못한 거 같아서 조금 미안하고 아쉬웠다. 프로젝트 수행 과정에서 AI 코드를 구현하는 부분, 즉 코드를 구체화하는 과정에서 어려움을 겪었는데 팀원의 아이디어와 노력 덕분에 잘 해결할 수 있었다. 또한 팀원들끼리 서로 짠 코드를 보고하고, 확인하고, 수정하는 과정에서 깃허브를 원활하게 활용하지 못한 거 같아서 그 부분이 아쉬웠다. 직접 게임을 만들어보는 팀 프로젝트는 처음이었는데 그 과정에서 교수님이 말씀하신 개인의 능력만이 아닌 팀원 전체의 의견 소통의 중요성을 알게 되었다. 좋은 팀원들을 만나서 성공적으로 잘 끝맺음해서 좋은 경험이 될 거 같다.

이다현: 팀원의 아이디어를 수용해 게임에 대한 검색을 통해 이해를 하여 게임에 필요한 시작창과 실행창 코드를 작성해보았고 다른 팀원들의 코드와 비교하면서 어떤 것이 더

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

좋은지 판단하고 게임을 짜는 코드를 조금 더 이해하고 더 좋은 방법을 모색하기 위해 생각했다. 다른 팀원들에 비해 아직까지 코드를 짜는 실력이 부족해 기능적이나 줄이는 부분에서 큰 도움을 주지 못 해서 죄송한 마음 뿐이었다. 특히 프로젝트를 진행 중 AI 코드를 짜는데에 있어서 많은 어려움을 겪었지만 팀원들의 노력으로 어려웠던 부분에서도 잘 해결할 수 있었다. 팀 프로젝트를 진행하면서 어렵고 힘든 부분도 많았지만 팀원들과 의견을 소통하면서 원활하게 진행이 되었고 좋은 경험과 많은 공부가 되었다.

임민석: 틱택토 시작창과 게임창을 전환하기 위해서 QStackedWidget 을 이용해 시작 코드를 짰으며, 혼자하기에서 사용되는 AI 를 구현하는 역할을 맡았음. AI 를 처음 구현하는 것이라 완성이 되기까지 수많은 시행착오가 있었음. 처음에는 각각의 상황을 하나하나 적으면서 반복했지만, 팀원들과 같이 게임 승리 조건을 알아내는 알고리즘을 만듦으로써 AI 의 지능 알고리즘을 완성시킬 수 있었음. 이 과정을 통해 프로젝트를 수행함에 있어 팀원 들과의 협동이 중요하다는 점을 느꼈으며, 협동을 하기 위해서는 활발한 소통이 이루어져야 한다는 사실을 알게 되었음. 깃허브를 통해 코드를 추가 및 수정했으면 버전 관리도 용이했을 것 같은데 이를 활용하지 못한 점이 아쉬움. 좋은 팀원들을 만난 덕분에 잘 마무리한 것 같아서 팀원들에게 매우 감사함.

4 참고 문헌

번호	종류	제목	출처	발행년도	저자	기타
1	유튜브	[파이썬]PYQT5 공부하기(28. 틱택토 게임 만들기 1/2)	https://www.youtube.com/watch?v=Ui90_Ux9Hwo	2018	파이썬 클래스	
2	유튜브	[파이썬]PYQT5 공부하기(29. 틱택토 게임 만들기 2/2)	https://youtu.be/SM3md_Ho44	2018	파이썬 클래스	

 소프트웨어융합대학 소프트웨어학부 소프트웨어프로젝트2	결과보고서	
	프로젝트 명	틱택토 게임 만들기
	팀 명	H조

3	블로그	[파이썬] 주피터노트북 아나콘다 pyinstaller 설치 및 사용법	https://dvlp-jun.tistory.com/26	2020	Dvlp_jun	
4	블로그	[다아라 개발기] 2. PyQt 에서 다중 레이아웃 넘나들기	https://hipolarbear.tistory.com/30	2021	Hi! Polarbear	

5 부록

사용자 매뉴얼

1. SW2 ADProject.zip 을 압축 해제한다.
2. StartWindow.exe 파일을 실행한다.
3. 자신이 원하는 게임 환경(혼자 또는 둘이 플레이할 것인지, 게임판의 크기는 어떻게 할 것인지)을 설정하고 Start 버튼을 누른다.
4. 재미있게 플레이한다.
5. 만약 게임 환경을 바꾸고 싶으면 프로그램을 끄고 2 번부터 다시 진행한다.