Научно-исследовательская работа

на тему

«Реализация игры Брейн-ринг с использованием библиотеки Pygame на Python 3»

Подготовил

Кондратенко Данила Александрович, 7а

Руководитель

Савченко Сергей Викторович, ИРООО, учитель информатики

Панченко Светлана Вячеславовна, учитель информатики и математики

БОУ «Полтавский лицей»

2017//2018 учебный год

Оглавление

Введение

Целеполагание

Цель работы: Изучить библиотеку Pygame и создать с её использованием программу для проведения Брейн-ринга.

Задачи:

1. Создать визуализацию счёта и таймера на мониторах.
2. Обеспечить возможность ведения счёта в игре.
3. Обеспечить возможность запуска и сброса таймера.
4. Обеспечить возможность остановки таймера дистанционно по нажатию кнопки одной из команд.
5. Обеспечить возможность обработки фальстартов.
6. Добавить звуковую сигнализацию моментов игры: запуска таймера, истечения времени, остановки таймера и фальстарта.

Определения

Python — объектно-ориентированный функциональный высокоуровневый язык программирования общего назначения с динамической типизацией, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций, в частности, обработка CSV[1] и JSON[2], статистические функции, сжатие файлов и многое другое. Всего в стандартной библиотеке насчитывается 242 модуля, из которых 12 связаны со средой исполнения Python 3, а 17 из них предоставляют специфичные для платформ функции. Он более приспособлен к работе с различными контейнерами (списками, кортежами, множествами, словарями), ведь в цикле for можно перебирать не индексы элементов, а сами элементы.

Язык был создан Гвидо ван Россумом, а первая версия (Python 0.9) вышла 20 февраля 1991. Название языка произошло, вопреки расхожему мифу, от названия юмористической программы 70-х годов «Monty Python's flying circus[3]». Позже это название было связано с названием змеи, но ошибочно, ведь в справочной документации можно заметить понятия, связанные с Monty Python или получившие популярность благодаря Monty Python, например, наряду с традиционными названиями foo и bar используются spam и eggs, также «spam» используется в выводимых результатах. Также язык содержит свою философию, именуемую «Дзен Пайтона» (The Zen of Python), выраженную в 19 предложениях.

1 CSV — **C**omma **s**eparated **v**alues (Значения, разделённые запятыми)
2 JSON — **J**ava**s**cript **o**bject **n**otation (Запись объектов Javascript)
3 Летающий цирк Монти Пайтона

Третья версия языка Python вышла 3 декабря 2008 года, получившая множество прогрессивных изменений по сравнению с Python 2, в частности, оператор print стал функцией, появились генераторы, убрано ограничение на размер целого числа в памяти, а кодировка строк была заменена на UTF-8[4].

Pygame (от **Py**thon и **game** - игра) — набор модулей для кроссплатформенной разработки игр. Базируется на мультимедийной библиотеке SDL[5]. Был создан Питом Шиннерсом, первая версия вышла 28 октября 2000 года. Последняя стабильная версия вышла 16 января 2017 года (Pygame 1.9.3)

Описание игры «Брейн-ринг»

Брейн-ринг — интеллектуальная игра, созданная Владимиром Ворошиловым по образу и подобию «Что? Где? Когда?», только в отличие от неё, одновременно соревнуются между собой две команды. Со временем «Брейн-ринг» стал одним из видов интеллектуального спорта, и по нему стали проводиться целые турниры.

В игре могут принимать участие 2 (в некоторых случаях — больше) команды, у каждой из которых есть одна кнопка. Ведущий задаёт вопрос и запускает таймер. Когда команда нажимает кнопку, происходит остановка таймера. В случае неправильного ответа ведущий запускает таймер снова, а если та или иная команда даёт правильный ответ, то ведущий обнуляет таймер и добавляет очки команде. Если на один вопрос не было дано правильного ответа командами, то за правильный ответ начисляется 2 очка. Если на два вопроса подряд не было дано правильного ответа, то 3 очка, и так далее. Иначе правильно ответившей команде присуждается 1 очко.

4 UTF-8 — **U**nicode **t**ransformation **f**ormat, **8**-bit (Восьмибитный формат трансформации Юникода)
5 SDL — **S**imple **D**irectMedia **l**ayer (Простой слой DirectMedia)

Процесс реализации

Создание визуализации счёта и таймера

Счёт

```
"""
Срез кода программы Брейн-ринг для счёта
"""
# Импортируем модули sys и pygame
import sys, pygame

# Создаём переменные для красной и зелёной команды
score_red, score_green = 0, 0
is_red = True

# Задаём цвета
black = (0, 0, 0)
white = (255, 255, 255)
red   = (255, 0, 0)
green = (0, 255, 0)

# Инициализируем экран и шрифты
pygame.font.init()
pygame.display.init()

# Задаём шрифт
score_font = pygame.font.SysFont(None, 400)

# Задаём окна
lscreen = pygame.Surface((640, 480)) # левая половина окна (счёт)
screen = pygame.display.set_mode((1280, 480)), pygame.FULLSCREEN) #
инициализируем окно 1280 на 480 пикселей и выводим его на весь экран

pygame.event.set_allowed([pygame.MOUSEBUTTONDOWN, pygame.KEYDOWN]) # Разрешаем
обработку событий мыши и клавиатуры

r1 = pygame.Rect(310, 230, 20, 20) # добавляем квадрат, показывающий, какой
команде присуждаются баллы

while True:
      lscreen.fill(black) # заполняем левую половину экрана чёрным
      # Вывод квадрата r1, окрашенного в цвет команды, которой нужно присудить
баллы
      if (is_red):
            pygame.draw.rect(lscreen, red, r1)
      else:
            pygame.draw.rect(lscreen, green, r1)
      # Перебираем произошедшие события
      for event in pygame.event.get():
            # Переключение команды (Numpad /)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_DIVIDE):
                  is_red = not is_red
            # Добавление очков команде (Numpad +)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_PLUS) and (is_red):
                  score_red += 1
```

```
            elif (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_PLUS) and not (is_red):
                  score_green += 1
            # Обнуление очков (Backspace)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_BACKSPACE):
                  score_red = 0
                  score_green = 0
            # Выход из программы
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_ESCAPE):
                  pygame.display.quit()
                  pygame.font.quit()
                  sys.exit(0)
      # Отрисовка
      ## Создаём объекты red_score и green_score, в которых хранится рисунок
числом, соответствующим счёту команд
      red_score = score_font.render(("%d" % score_red), True, red)
      green_score = score_font.render(("%d" % score_green), True, green)
      ## Перерисовываем эти объекты на левую половину окна, центруя их
      lscreen.blit(red_score, [(lscreen.get_width()//2 -
red_score.get_width()) // 2, (lscreen.get_height() - red_score.get_height()) //
2])
      lscreen.blit(green_score, [((lscreen.get_width()//2 -
green_score.get_width()) // 2)+lscreen.get_width()//2, (lscreen.get_height() -
green_score.get_height()) // 2])
      ## Перерисовываем левую половину окна
      screen.blit(lscreen, [0, 0])
      ## Обновляем дисплей
      pygame.display.update()
```

Таймер

```
"""
Срез кода программы Брейн-ринг для таймера
"""
# Импортируем необходимые модули
import sys, pygame, threading
import pygame.mixer
from xTimer import XTimer # библиотека XTimer по адресу
https://github.com/ivanhalencp/python/tree/master/xTimer

# Задаём параметры
lock = False            # блокировка кнопки
timer_tm = False        # работа таймера
cmd = 0                 # номер команды: 1, 3 - красная; 2, 4 - зелёная

# Задаём переменные времени
current = 0.00
end = 60.00             # длительность работы таймера до истечения времени

current_str = None

# Инициализируем звуки
pygame.mixer.init()

# Задаём звуковые сигналы
start_snd = pygame.mixer.Sound('timer_start.wav')          # запуск таймера
stop_snd = pygame.mixer.Sound('timer_stop.wav')            # остановка таймера
end_snd = pygame.mixer.Sound('timer_end.wav')              # истечение времени
falstart_snd = pygame.mixer.Sound('timer_falstart.wav')    # фальстарт
```

```
# Задаём функции timeout_timer и play_end, необходимые для потоков
## Изменение значения таймера
def timeout_timer():
      global current, end                # Без этого функция не попрёт, ибо
current и end - переменные глобальные
      if (current <= end):
            current = current + 0.1

## Проигрывание звука окончания
def play_end():
      global current, end, timer_tm      # Глобальные переменные
      while True:
            if (current >= end) and (timer_tm):
                  end_snd.play()          # Проигрывание сигнала окончания
                  timer_tm = False        # Установка состояния таймера

# Задаём цвета
black = (0, 0, 0)
white = (255, 255, 255)
red   = (255, 0, 0)
green = (0, 255, 0)

# Инициализируем экран и шрифты
pygame.font.init()
pygame.display.init()

# Задаём шрифт
timer_font = pygame.font.SysFont('Courier New', 250)

# Создаём таймер
timer = XTimer(0.1, timeout_timer)
timer.init()            # инициализация
# Нижеследующие две строчки нужны для того, чтобы таймер был всегда наготове
timer.start()           # запуск
timer.pause()           # пауза

# Создаём поток check, в котором будет постоянно проверяться, нужно ли будет
подавать сигнал о истечении времени
check = threading.Thread(target=play_end)
check.start()

# Задаём окна
rscreen = pygame.Surface((640, 480)) # правая половина окна (таймер)
screen = pygame.display.set_mode((1280, 480)), pygame.FULLSCREEN) #
инициализируем окно 1280 на 480 пикселей и выводим его на весь экран

pygame.event.set_allowed([pygame.MOUSEBUTTONDOWN, pygame.KEYDOWN]) # Разрешаем
обработку событий мыши и клавиатуры

while True:
      current_str = '{:.1f}'.format(current)   # Строка, соответствующая
значению таймера
      # Перебираем произошедшие события
      for event in pygame.event.get():
            # Запуск таймера
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_RETURN):
                  cmd = 0                      # Сброс номера команды
                  pygame.mixer.stop()          # Остановка всех звуков
                  lock = False                 # Снятие блокировки
                  timer.start()                # Запуск таймера (2 строки)
```

```
                timer.run()
                timer_tm = True                 # Установка состояния таймера
                start_snd.play()                # Проигрывание сигнала запуска
            # Остановка и обнуление таймера
            if (event.type == pygame.KEYDOWN) and (event.key == pygame.K_SPACE):
                cmd = 0                         # Сброс номера команды
                pygame.mixer.stop()             # Остановка всех звуков
                lock = False                    # Снятие блокировки
                timer.terminate()               # Остановка таймера
                timer_tm = False                # Установка состояния таймера
                current = 0.00                  # Обнуление таймера
            # Остановка красной командой (правая кнопка мыши)
            if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 3)
and (not lock):
                # Остановка во время работы таймера
                if (timer_tm):
                    cmd = 1                     # Установка номера команды
                    timer.pause()               # Приостановка таймера
                    pygame.mixer.stop()         # Остановка всех звуков
                    stop_snd.play()             # Проигрывание сигнала остановки
                    lock = True                 # Установка блокировки
                # Фальстарт
                else:
                    cmd = 3                     # Установка номера команды
                    timer.terminate()           # Остановка таймера
                    pygame.mixer.stop()         # Остановка всех звуков
                    falstart_snd.play()         # Проигрывание звука фальстарта
                    break
            # Остановка зелёной командой (средняя кнопка мыши)
            if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 2)
and (not lock):
                # Остановка во время работы таймера
                if (timer_tm):
                    cmd = 2                     # Установка номера команды
                    timer.pause()               # Приостановка таймера
                    pygame.mixer.stop()         # Остановка всех звуков
                    stop_snd.play()             # Проигрывание сигнала остановки
                    lock = True                 # Установка блокировки
                # Фальстарт
                else:
                    cmd = 4                     # Установка номера команды
                    timer.terminate()           # Остановка таймера
                    pygame.mixer.stop()         # Остановка всех звуков
                    falstart_snd.play()         # Проигрывание звука фальстарта
                    break
            # Выход из программы
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_ESCAPE):
                pygame.mixer.quit()
                pygame.display.quit()
                pygame.font.quit()
                sys.exit(0)
        # Окраска фона
        ## Команда красная, ответ во время работы таймера
        if (cmd == 1):
            rscreen.fill(red)
        ## Команда зелёная, ответ во время работы таймера
        elif (cmd == 2):
            rscreen.fill(green)
        ## Стандартный фон
        else:
```

```
            rscreen.fill(black)
      # Окраска текста
      # Создаём объект zeit, в котором будет храниться рисунок, точно
соответствующий значению на таймере
      ## Команда красная, фальстарт
      if (cmd == 3):
            zeit = timer_font.render(current_str, True, red)
      ## Команда зелёная, фальстарт
      elif (cmd == 4):
            zeit = timer_font.render(current_str, True, green)
      ## Стандартный цвет текста
      else:
            zeit = timer_font.render(current_str, True, white)
      # Отрисовка
      ## Перерисовываем на правую половину окна объект zeit, центруя его
      rscreen.blit(zeit, [(rscreen.get_width()-zeit.get_width())//2,
(rscreen.get_height()-zeit.get_height())//2])
      ## Перерисовываем правую половину окна
      screen.blit(rscreen, [(screen.get_width())//2, 0])
      ## Обновляем дисплей
      pygame.display.update()
```

Необходимое оборудование

Вывод изображения

Чтобы зрители и участники могли наблюдать за ходом игры, было принято решение выводить изображение на два монитора, ведь проекторы не всегда отображают изображение в приемлемом качестве.

Кнопки

При реализации игры «Брейн-ринг» возник вопрос о кнопках, необходимых для каждой команды. Было принято решение дистанционного соединения кнопок с компьютером. В качестве беспроводных кнопок использовалась технология и принцип действия беспроводных мышек. Так как предусмотрена возможность игры двух команд, то необходимо две кнопки.

Заключение

Таким образом, мы можем считать, что наша цель достигнута. На опыте мы выяснили, что библиотека Pygame, с помощью которой осуществлялась реализация игры «Брейн-ринг», не только удовлетворяет наши требования, но и пригодна к использованию для реализации более сложных проектов. С помощью неё можно создавать не только игры. Она может использоваться и для визуализации математических графиков и технических процессов.

Полный код игры «Брейн-ринг» с комментариями

```
# Импортируем необходимые модули
import sys, pygame, threading
import pygame.mixer
from xTimer import XTimer # библиотека XTimer по адресу
https://github.com/ivanhalencp/python/tree/master/xTimer

# Задаём параметры
lock = False            # блокировка кнопки
timer_tm = False        # работа таймера
cmd = 0                 # номер команды: 1, 3 - красная; 2, 4 - зелёная

# Задаём переменные времени
current = 0.00
end = 60.00             # длительность работы таймера до истечения времени

current_str = None

# Создаём переменные для красной и зелёной команды
score_red, score_green = 0, 0
is_red = True

# Инициализируем звуки
pygame.mixer.init()

# Задаём звуковые сигналы
start_snd = pygame.mixer.Sound('timer_start.wav')          # запуск таймера
stop_snd = pygame.mixer.Sound('timer_stop.wav')            # остановка таймера
end_snd = pygame.mixer.Sound('timer_end.wav')              # истечение времени
falstart_snd = pygame.mixer.Sound('timer_falstart.wav')    # фальстарт

# Задаём функции timeout_timer и play_end, необходимые для потоков
## Изменение значения таймера
def timeout_timer():
      global current, end                       # Без этого функция не попрёт,
ибо current и end - переменные глобальные
      if (current <= end):
            current = current + 0.1

## Проигрывание звука окончания
def play_end():
      global current, end, timer_tm # Глобальные переменные
      while True:
            if (current >= end) and (timer_tm):
                  end_snd.play()                # Проигрывание сигнала окончания
                  timer_tm = False              # Установка состояния таймера

# Задаём цвета
black = (0, 0, 0)
white = (255, 255, 255)
red   = (255, 0, 0)
green = (0, 255, 0)

# Инициализируем экран и шрифты
pygame.font.init()
pygame.display.init()
```

```
# Задаём шрифты
score_font = pygame.font.SysFont(None, 400)
timer_font = pygame.font.SysFont('Courier New', 250)

# Создаём таймер
timer = XTimer(0.1, timeout_timer)
timer.init()            # инициализация
# Нижеследующие две строчки нужны для того, чтобы таймер был всегда наготове
timer.start()           # запуск
timer.pause()           # пауза

# Создаём поток check, в котором будет постоянно проверяться, нужно ли будет
подавать сигнал о истечении времени
check = threading.Thread(target=play_end)
check.start()

# Задаём окна
lscreen = pygame.Surface((640, 480)) # левая половина окна (счёт)
rscreen = pygame.Surface((640, 480)) # правая половина окна (таймер)
screen = pygame.display.set_mode((1280, 480)), pygame.FULLSCREEN) #
инициализируем окно 1280 на 480 пикселей и выводим его на весь экран

pygame.event.set_allowed([pygame.MOUSEBUTTONDOWN, pygame.KEYDOWN]) # Разрешаем
обработку событий мыши и клавиатуры

r1 = pygame.Rect(310, 230, 20, 20)

while True:
      current_str = '{:.1f}'.format(current)    # Строка, соответствующая
значению таймера
      # Перебираем произошедшие события
      for event in pygame.event.get():
            # Запуск таймера (Enter)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_RETURN):
                  cmd = 0                       # Сброс номера команды
                  pygame.mixer.stop()           # Остановка всех звуков
                  lock = False                  # Снятие блокировки
                  timer.start()                 # Запуск таймера (2 строки)
                  timer.run()
                  timer_tm = True               # Установка состояния таймера
                  start_snd.play()        # Проигрывание сигнала запуска
            # Остановка и обнуление таймера (Space)
            if (event.type == pygame.KEYDOWN) and (event.key == pygame.K_SPACE):
                  cmd = 0                       # Сброс номера команды
                  pygame.mixer.stop()           # Остановка всех звуков
                  lock = False                  # Снятие блокировки
                  timer.terminate()             # Остановка таймера
                  timer_tm = False              # Установка состояния таймера
                  current = 0.00                # Обнуление таймера
            # Остановка красной командой (правая кнопка мыши)
            if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 3)
and (not lock):
                  # Остановка во время работы таймера
                  if (timer_tm):
                        cmd = 1                 # Установка номера команды
                        timer.pause()           # Приостановка таймера
                        pygame.mixer.stop()     # Остановка всех звуков
                        stop_snd.play()         # Проигрывание сигнала остановки
                        lock = True             # Установка блокировки
                  # Фальстарт
```

```
                  else:
                        cmd = 3                 # Установка номера команды
                        timer.terminate()       # Остановка таймера
                        pygame.mixer.stop()     # Остановка всех звуков
                        falstart_snd.play()     # Проигрывание звука фальстарта
                        break
            # Остановка зелёной командой (средняя кнопка мыши)
            if (event.type == pygame.MOUSEBUTTONDOWN) and (event.button == 2)
and (not lock):
                  # Остановка во время работы таймера
                  if (timer_tm):
                        cmd = 2                 # Установка номера команды
                        timer.pause()           # Приостановка таймера
                        pygame.mixer.stop()     # Остановка всех звуков
                        stop_snd.play()         # Проигрывание сигнала остановки
                        lock = True             # Установка блокировки
                  # Фальстарт
                  else:
                        cmd = 4                 # Установка номера команды
                        timer.terminate()       # Остановка таймера
                        pygame.mixer.stop()     # Остановка всех звуков
                        falstart_snd.play()     # Проигрывание звука фальстарта
                        break
            # Переключение команды (Numpad /)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_DIVIDE):
                  is_red = not is_red
            # Добавление очков команде (Numpad +)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_PLUS) and (is_red):
                  score_red += 1
            elif (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_KP_PLUS) and not (is_red):
                  score_green += 1
            # Обнуление очков (Backspace)
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_BACKSPACE):
                  score_red = 0
                  score_green = 0
            # Выход из программы
            if (event.type == pygame.KEYDOWN) and (event.key ==
pygame.K_ESCAPE):
                  pygame.mixer.quit()
                  pygame.display.quit()
                  pygame.font.quit()
                  sys.exit(0)
      # Окраска фона
      ## Команда красная, ответ во время работы таймера
      if (cmd == 1):
            rscreen.fill(red)
      ## Команда зелёная, ответ во время работы таймера
      elif (cmd == 2):
            rscreen.fill(green)
      ## Стандартный фон
      else:
            rscreen.fill(black)
      # Окраска текста
      # Создаём объект zeit, в котором будет храниться рисунок, точно
соответствующий значению на таймере
      ## Команда красная, фальстарт
      if (cmd == 3):
```

```
            zeit = timer_font.render(current_str, True, red)
      ## Команда зелёная, фальстарт
      elif (cmd == 4):
            zeit = timer_font.render(current_str, True, green)
      ## Стандартный цвет текста
      else:
            zeit = timer_font.render(current_str, True, white)
      # Отрисовка
      ## Создаём объекты red_score и green_score, в которых хранится рисунок
числом, соответствующим счёту команд
      red_score = score_font.render(("%d" % score_red), True, red)
      green_score = score_font.render(("%d" % score_green), True, green)
      ## Перерисовываем эти объекты на левую полвину окна, центруя их
      lscreen.blit(red_score, [(lscreen.get_width()//2 -
red_score.get_width()) // 2, (lscreen.get_height() - red_score.get_height()) //
2])
      lscreen.blit(green_score, [((lscreen.get_width()//2 -
green_score.get_width()) // 2)+lscreen.get_width()//2, (lscreen.get_height() -
green_score.get_height()) // 2])
      ## Перерисовываем на правую половину окна объект zeit, центруя его
      rscreen.blit(zeit, [(rscreen.get_width()-zeit.get_width())//2,
(rscreen.get_height()-zeit.get_height())//2])
      ## Перерисовываем левую половину окна
      screen.blit(lscreen, [0, 0])
      ## Перерисовываем правую половину окна
      screen.blit(rscreen, [(screen.get_width())//2, 0])
      ## Обновляем дисплей
      pygame.display.update()
```

Клавиша	Назначение
`Enter`	Запуск таймера
`Space`	Обнуление таймера
`Numpad /`	Переключение команды
`Numpad +`	Добавление очков команде
`Backspace`	Обнуление команды
`Escape`	Выход
Правая кнопка мыши	Остановка красной командой
Средняя кнопка мыши	Остановка зелёной командой

Приложение 2

Правила игры «Брейн-ринг»

1. Общие положения

   1.1. Данный документ описывает правила игры "Брейн-ринг", а также определяет некоторые общепринятые термины. 1.2. Соревнования по "Брейн-рингу" состоят из матчевых встреч (далее - «бои»). В каждом бое участвуют 2 команды, если иное не установлено регламентом соревнования (далее - "регламент"). Ниже под командами везде имеются в виду команды, принимающие участие в текущем бое. 1.3. Во время боя в составе команды может находиться не более 6 человек. Замены по ходу боя запрещены, если иное не установлено регламентом.

2. Проведение боя

   2.1. Каждый бой состоит из одного или нескольких вопросных раундов. В каждом вопросном раунде задается один вопрос.

   2.2. Вопросы "Брейн-ринга" должны соответствовать Кодексу спортивного "Что? Где? Когда?", если иное не установлено регламентом.

   2.3. Регламент соревнования должен устанавливать количество вопросных раундов в бое или условия окончания боя.

   2.4. Цель игры состоит в том, чтобы дать правильный ответ на вопрос строго в отведённое время. Команды могут давать ответы по очереди, но не одновременно. В течение вопросного раунда команда может дать не более одного ответа.

   2.5. Вопросы задаются ведущим. Он же оценивает правильность ответов команд.

   2.6. Определение правильности ответов производится в соответствии с Кодексом спортивного "Что? Где? Когда", если иное не установлено регламентом.

   2.7. Если регламент разрешает подачу апелляций на решения ведущего, то он должен содержать полное описание процедур подачи и рассмотрения апелляций.

   2.8. Бой считается завершённым, когда ведущий объявляет его результат.

3. Определение результатов боя

   3.1. За правильный ответ давшая его команда получает опредёленное количество игровых очков (далее - "очки"). Наиболее распространённые схемы начисления очков описаны в двух следующих пунктах.

   3.2. При игре по схеме без накопления за каждый правильный ответ команда получает 1 очко.

   3.3. При игре по схеме с накоплением используются следующие правила:

      - за правильный ответ на первый вопрос боя команда получает 1 очко;

- количество очков, начисляемое за правильный ответ в последующих вопросных раундах, зависит от исхода одного или нескольких предыдущих раундов;
- если в предыдущем раунде был дан правильный ответ, то в текущем раунде за правильный ответ начисляется 1 очко;
- если в предыдущем раунде ни одна команда не дала правильного ответа, то в текущем раунде за правильный ответ начисляется на 1 очко больше, чем было бы начислено за правильный ответ в предыдущем раунде.

Схему начисления очков должен определять регламент.

3.4. Победительницей боя объявляется команда, набравшая по сумме всех вопросных раундов большее количество очков. Если в бое участвует более 2 команд, они распределяются по занятым в бое местам в соответствии с количеством набранных за все вопросные раунды очков.

3.5. В случае равенства очков у обеих команд бой считается закончившимся вничью. Если в бое участвует более 2 команд, то в любой паре команд команды с одинаковым количеством очков делят соответствующие места. Регламент может устанавливать способ определения победителя боя (и/или последующих мест) при равенстве очков.

4. Брейн-система

4.1. Для определения команды, получающей право ответа на вопрос, используется специальное устройство (далее - "брейн-система"). Функции брейн-системы описаны в последующих пунктах.

4.2. Брейн-система должна подавать световой и/или звуковой сигнал, означающий начало времени вопросного раунда, отведённого для ответов (далее начало отведённого для ответов времени называется "запуском брейн-системы").

4.3. Брейн-система должна подавать световой и/или звуковой сигнал, означающий окончание времени вопросного раунда, отведённого для ответов. Желательно, чтобы брейн-система издавала звуковой сигнал раз в секунду в последние 5 секунд отведённого для ответов времени. В ином случае отсчёт последних 5 секунд должен производить ведущий или его ассистент.

4.4. Брейн-система должна давать командам возможность сигнализировать ведущему о готовности дать ответ. Как правило, эта функция реализуется в виде периферийных устройств брейн-системы, располагающихся по одному у каждой команды (далее — "кнопок").

4.5. В случае когда одна из команд просигнализировала о готовности дать ответ (далее - "нажала на кнопку"), брейн-система должна блокировать сигналы от другой команды (или от всех других команд) до соответствующего действия ведущего или его ассистента.

4.6. Брейн-система должна отсчитывать время, отведённое для ответов на вопрос. Когда одна из команд нажимает на кнопку, брейн-система должна останавливать отсчёт времени до соответствующего действия ведущего или его ассистента.

4.7. Брейн-система должна давать ведущему или его ассистенту чёткую возможность определить, была ли кнопка нажата в отведённое для ответов время, до его начала или после его окончания.

5. Проведение вопросного раунда

5.1. Перед началом вопросного раунда ведущий или его ассистент приводит брейн-систему в исходное состояние, при котором отсчёт времени не производится, но нажатия кнопок регистрируются.

5.2. Началом вопросного раунда являются слова ведущего "вопрос номер", после которых ведущий объявляет номер вопроса в текущем бою.

5.3. После объявления номера вопроса ведущий зачитывает сам вопрос. Когда чтение вопроса окончено, ведущий произносит слово "время", после чего ведущий или его ассистент запускает брейн-систему. Промежуток между словом "время" и запуском брейн-системы не должен превышать 3 секунд.

5.4. В начале вопросного раунда обе (или все) команды имеют право ответа на вопрос текущего раунда.

5.5. Если команда нажала на кнопку после того, как ведущий объявил номер вопроса, но до запуска брейн-системы (далее такая ситуация называется "фальстарт"), то команда лишается права отвечать на текущий вопрос.

5.6. После запуска брейн-системы начинается отсчёт отведённого на ответы времени. Это время равно 60 секундам. Регламент может устанавливать, что в случае фальстарта отвёденное на ответы время уменьшается.

5.7. Если команда по ходу отвёденного для ответов времени нажала на кнопку, она получает исключительное право дать ответ на вопрос. При этом ведущий или его ассистент останавливает отсчёт отведённого времени на брейн-системе и переводит систему в состояние, при котором она лишь регистрирует нажатия на кнопку других команд. Далее ведущий указывает команду, получившую право дать ответ.

5.8. Команда, получившая право дать ответ, должна немедленно определить одного игрока, дающего ответ (далее - отвечающего игрока). Капитан команды может назвать ведущему отвечающего игрока, но не обязан это делать, если иное не установлено регламентом. Отвечающий игрок должен немедленно дать ответ. Остальные игроки команды должны хранить молчание до окончания ответа и не имеют права каким-либо образом подсказывать отвечающему игроку.

5.9. Если команда нарушает какое-либо из правил пункта 5.8, то её ответ автоматически признаётся неправильным. Факт нарушения правил устанавливается ведущим.

5.10. Если команда дает правильный ответ, вопросный раунд заканчивается.

5.11. Если команда дает неправильный ответ, она лишается права ответа в текущем раунде. Если обе (или все) команды лишились права ответа, вопросный раунд заканчивается.

5.12. Если после неправильного ответа одна команда (или несколько команд) сохраняет право ответа, ведущий или его ассистент возобновляет отсчёт времени на брейн-системе (далее такая команда называется «перезапуском брейн-системы»).

5.13. Если после неправильного ответа одной из команд право на ответ сохранило несколько команд и одна из этих команд нажала на кнопку до перезапуска брейн-системы, фиксируется, и данная команда теряет право ответа. Если право ответа сохранила только одна команда, фальстарт не фиксируется.

5.14. После перезапуска брейн-системы возможны два варианта отсчёта отведённого на ответы времени:

- отсчёт продолжается с текущего значения;
- устанавливается новое значение отведённого на ответы времени (например команда, сохранившая право ответа, получает 20 секунд).

Правило отсчёта времени после перезапуска системы должен определять регламент.

5.15. Если отведённое на ответы время истекло, вопросный раунд заканчивается.

5.16. Если ни одна команда не дала правильный ответ, ведущий объявляет его сразу же после окончания вопросного раунда. Возможна разновидность этого правила, когда ведущий не объявляет правильный ответ и задаёт тот же самый вопрос в следующем раунде. В этом случае регламент должен в явном виде определять, сколько раундов подряд может играться один и тот же вопрос, а правильный ответ на вопрос должен быть объявлен до завершения боя.