Створення прототипу мобільної гри на основі фреймворку Kivy

1. Загальні відомості

Kivy — це бібліотека Python із відкритим вихідним кодом для розробки мобільних застосунків з підтримкою мультитач та іншого прикладного програмного забезпечення з природнім інтерфейсом користувача (Natural user interface, NUI). Вона може працювати під управлінням Android, IOS, GNU/Linux, OS X і Windows. Поширюється на умовах ліцензії MIT. Kivy є вільним ПЗ із відкритим вихідним кодом.

Kivy містить всі елементи для створення прикладної програми, такі як:

– розширена підтримка введення для миші, клавіатури, TUIO (Tangible user interface) і ОС-специфічних мультитач подій;

– графічна бібліотека з використанням тільки OpenGL ES 2, на основі Vertex Buffer Object і шейдерів;

– широкий спектр віджетів з підтримкою мультитач;

– проміжна мова (Kv), що використовується для легкого створення власних віджетів [1].

2. Заготовка застосунку Kivy

Розглянемо структуру застосунку Kivy на прикладі гри Pong (симулятор настільного тенісу). Наприкінці ми одержимо мінімальну придатну до гри версію Pong. За основу було взято навчальний приклад, запропонований офіційним сайтом Kivy [2]. Для виконання наведених нижче прикладів має бути встановлений інтерпретатор Python і бібліотека Kivy. Інструкція зі встановлення доступна на офіційному сайті [3].

Створіть новий проект Python (ця дія відрізнятиметься в різних середовищах програмування). Створіть головний файл проекту, наприклад, main.py, і впишіть у нього цей код:

```
from kivy.app import App
from kivy.uix.widget import Widget

class PongGame(Widget):
    pass

class PongApp(App):
    def build(self):
        return PongGame()

if __name__ == '__main__':
    PongApp().run()
```

В перших двох рядках імпортується клас застосунку Kivy `App` і клас абстрактного віджету `Widget`. Створення застосунку Kivy передбачає створення класу-нащадку від `App` (в нашому прикладі клас `PongApp`) та реалізація в ньому методу `build()`, що має повертати екземпляр головного віджету (в нашому випадку це віджет `PongGame`, що наслідується від `Widget`). Всі віджети утворюють дерево, в корені якого знаходиться головний або кореневий віджет застосунку. Запустіть програму. Якщо все зроблено вірно, виведеться пусте вікно: наш віджет ще ніяк не відображає себе на екрані.

▶Оператор `pass` в Python — це пустий оператор, "заглушка", яку пізніше замінить код класу.

Для налаштування зовнішнього вигляду класу `PongGame` створіть файл pong.kv з таким вмістом:

```
#:kivy 1.0.9

<PongGame>:
    canvas:
        Rectangle:
            pos: self.center_x - 5, 0
            size: 10, self.height

    Label:
        font_size: 70
        center_x: root.width / 4
        top: root.top - 50
        text: "0"

    Label:
        font_size: 70
        center_x: root.width * 3 / 4
        top: root.top - 50
        text: "0"
```

Файли .kv — це файли, написані вбудованою мовою *KV language*, що

описують дерево віджетів у декларативній формі. Перший рядок — це заголовок kv-файлу. В ньому вказується мінімальна необхідна версія Kivy. Нижче знаходиться заголовок `<PongGame>:`, який означає, що подальший код стосуватиметься класу `PongGame`. Як і в Python, в мові KV застосовуються відступи для вбудованих блоків. Блок, що йде за іменем класу в дужках <>, називається *правилом* для цього віджету. Це правило буде застосовуватися для всіх екземплярів класу. Всередині правила можна:

– встановлювати значення полів;

– додавати дочірні віджети;

– вставляти секцію `canvas`, в якій знаходяться інструкції з малювання віджетау.

В нашому прикладі код секції `canvas` вказує на те, що віджет має малювати прямокутник, ліва верхня вершина якого зміщена на 5 пікселів вліво від геометричного центру віджету й має координату 0 по осі y. Прямокутник має ширину 10 пікселів, а його висота дорівнює висоті віджету. Цей прямокутник позначатиме середину ігрового поля.

► Увага! Ім'я файлу .kv має відповідати імені класу застосунку. Оскільки ми назвали його PongApp, то файл має називатися pong.kv (назва застосунку без суфіксу App).

В нашому .kv файлі описано також два текстових поля для відображення рахунку гри. Зараз вони відображують значення "0". Зверніть увагу, що всі координати вказуються відносно координат самого віджету. В секції `canvas` доступ до них одержується за допомогою префікса `self`, а в дочірніх віджетах за допомогою префікса `root` (для текстового поля наш віджет є кореневим).

Запустіть проект. Змінюйте розміри вікна й спостерігайте, як поводяться `canvas` та дочірні віджети. Використання відносних координат гарантує прийнятний вигляд нашого застосунку на різних екранах.

3. Створення віджету PongBall

Ми вже маємо ігрове поле. Тепер час реалізувати віджет, що відображатиме м'яч. Додайте наведені нижче рядки в головний файл проекту:

```
class PongBall(Widget):

    # швидкість м'яча по осі x та y
    velocity_x = NumericProperty(0)
    velocity_y = NumericProperty(0)

    # ReferenceListProperty створює поле, що являє
    # собою кортеж з інших полів класу
    velocity = ReferenceListProperty(velocity_x, velocity_y)

    # Функція «move» переміщує м'яч на один крок.
    # Вона викликатиметься через рівні проміжки часу
    def move(self):
        self.pos = Vector(*self.velocity) + self.pos
```

В файл pong.kv додайте правило для класу PongBall:

```
<PongBall>:
    size: 50, 50
    canvas:
        Ellipse:
            pos: self.pos
            size: self.size
```

і в секцію PongGame вставте дочірній віджет PongBall:

```
<PongGame>:
...
    PongBall:
        center: self.parent.center
```

Також на початку годовного модуля підключіть необхідні класи з модулів `kivy.properties` та `kivy.vector`:

```
from kivy.properties import NumericProperty, ReferenceListProperty
from kivy.vector import Vector
```

Запустіть проект і переконайтеся, що м'яч відображується на ігровому полі.

Тепер змусимо наш м'яч рухатися. В Kivy є клас `Clock`, що дозволяє викликати певну функцію через рівні проміжки часу. Підключіть відповідний модуль в головному файлі проекту:

```
from kivy.clock import Clock
```

Створимо в класі `PongGame` метод `update()`, що буде оновлювати кореневий віджет і буде викликатися за таймером. Зв'язування з таймером реалізуємо за допомогою `Clock.schedule_interval()` у методі `build()` класу `PongApp`, тобто одразу після запуску програми. Змініть класи `PongGame` та `PongApp` таким чином:

```
class PongGame(Widget):

    def update(self, dt):
        # пізніше тут буде переміщення м'яча
        pass

class PongApp(App):

    def build(self):
        game = PongGame()
        Clock.schedule_interval(game.update, 1.0/60.0)
        return game
```

Поки що ми не можемо звернутися до віджета-м'яча, позаяк він створений не в коді Python, а засобами мови KV. Щоб одержати доступ до таких віджетів, можна використати функцію `ObjectProperty()`. Щоб нею скористатися, спочатку імпортуйте її з модулю kivy.properties в головному файлі проекту:

```
from kivy.properties import NumericProperty, ReferenceListProperty, ObjectProperty
```

Тепер можна створити в класі PongGame властивість ball та реалізувати функцію руху м'яча. Змініть код класу PongGame таким чином:

```
class PongGame(Widget):
    ball = ObjectProperty(None)

    def update(self, dt):
        self.ball.move()

        # відбивання верхньої й нижньої границі
        if (self.ball.y < 0) or (self.ball.top > self.height):
            self.ball.velocity_y *= -1

        # відбивання від правої й лівої границі
        if (self.ball.x < 0) or (self.ball.right > self.width):
            self.ball.velocity_x *= -1
```

Перш, ніж запустити проект, додайте в .kv файлі id для дочірнього віджета й зв'яжіть цей id із властивістю `ball`:

```
<PongGame>:
    ball: pong_ball

    # ... (canvas and Labels)

    PongBall:
        id: pong_ball
        center: self.parent.center
```

Лишилося додати функцію serve_ball(), що встановлює випадкову початкову швидкість м'яча. На цьому етапі головний модуль виглядатиме так:

```
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.properties import NumericProperty, ReferenceListProperty, ObjectProperty
from kivy.vector import Vector
from kivy.clock import Clock
from random import randint

class PongBall(Widget):
    velocity_x = NumericProperty(0)
    velocity_y = NumericProperty(0)
    velocity = ReferenceListProperty(velocity_x, velocity_y)

    def move(self):
        self.pos = Vector(*self.velocity) + self.pos

class PongGame(Widget):
    ball = ObjectProperty(None)

    def serve_ball(self):
        self.ball.center = self.center
        self.ball.velocity = Vector(4, 0).rotate(randint(0, 360))

    def update(self, dt):
        self.ball.move()

        # відбивання верхньої й нижньої границі
        if (self.ball.y < 0) or (self.ball.top > self.height):
            self.ball.velocity_y *= -1

        # відбивання від правої й лівої границі
        if (self.ball.x < 0) or (self.ball.right > self.width):
            self.ball.velocity_x *= -1

class PongApp(App):
    def build(self):
        game = PongGame()
        game.serve_ball()
        Clock.schedule_interval(game.update, 1.0 / 60.0)
        return game

if __name__ == '__main__':
    PongApp().run()
```

При цьому ваш .kv файл матиме такий вигляд:

```
#:kivy 1.0.9

<PongBall>:
    size: 50, 50
    canvas:
        Ellipse:
            pos: self.pos
            size: self.size

<PongGame>:
    ball: pong_ball

    canvas:
        Rectangle:
            pos: self.center_x-5, 0
            size: 10, self.height

    Label:
        font_size: 70
        center_x: root.width / 4
        top: root.top - 50
        text: "0"

    Label:
        font_size: 70
        center_x: root.width * 3 / 4
        top: root.top - 50
        text: "0"

    PongBall:
        id: pong_ball
        center: self.parent.center
```

Запустіть проект і переконайтеся, що м'яч рухається коректно й відбивається від меж поля.

Тепер лишилося додати лише рухомі ракетки гравців та ігровий рахунок. Не будемо детально розглядати створення віджету й правил для нього в файлі .kv. Ця процедура описана вище на прикладі попередніх віджетів. Зупинимося на тому, як рухати віджет у відповідь на дії користувача. Найбільш типовими для роботи з сенсорним екраном є події `on_touch_down`, `on_touch_move` й `on_touch_up`. Їх назви говорять самі за себе. Для опрацювання цих подій слід реалізувати у віджеті функції з такими іменами. Якщо ви не опрацьовуєте ці події, віджет передає їх своїм дочірнім віджетам доки хтось із них не опрацює подію.

►При запуску на настільній системі події від сенсорного екрану будуть емулюватися мишею.

Правил Pong такі прості, що нема жодної потреби передавати події від сенсорного екрану віджетам-ракеткам. Опрацюємо їх у кореневому віджеті:

```
def on_touch_move(self, touch):
    if touch.x < self.width/3:
        self.player1.center_y = touch.y
    if touch.x > self.width - self.width/3:
        self.player2.center_y = touch.y
```

Якщо подія виникає в лівій третині кореневого віджета, то рухаємо ліву ракетку, якщо в правій третині — праву. Ракетка рухається лише вздовж напрямку y, тому змінюється лише одна координата.

Рахунок гри буде зберігатися в NumericProperty, механізм роботи якого схожий на ObjectProperty. В свою чергу, значення текстових полів Label у .kv файлі пов'язані з відповідними властивостями. Тому не треба буде оновлювати

їх вручну. Ракетка гравця реалізована в класі `PongPaddle`. Його метод `bounce_ball()` враховує, в яке місце ракетки влучає м'яч і коректує його швидкість. Остаточний код головного модулю має вигляд:

```
from kivy.app import App
from kivy.uix.widget import Widget
from kivy.properties import NumericProperty, ReferenceListProperty,\
    ObjectProperty
from kivy.vector import Vector
from kivy.clock import Clock

class PongPaddle(Widget):
    score = NumericProperty(0)

    def bounce_ball(self, ball):
        if self.collide_widget(ball):
            vx, vy = ball.velocity
            offset = (ball.center_y - self.center_y) / (self.height / 2)
            bounced = Vector(-1 * vx, vy)
            vel = bounced * 1.1
            ball.velocity = vel.x, vel.y + offset

class PongBall(Widget):
    velocity_x = NumericProperty(0)
    velocity_y = NumericProperty(0)
    velocity = ReferenceListProperty(velocity_x, velocity_y)

    def move(self):
        self.pos = Vector(*self.velocity) + self.pos

class PongGame(Widget):
    ball = ObjectProperty(None)
    player1 = ObjectProperty(None)
    player2 = ObjectProperty(None)

    def serve_ball(self, vel=(4, 0)):
        self.ball.center = self.center
        self.ball.velocity = vel

    def update(self, dt):
        self.ball.move()

        # відбивання від ракеток
        self.player1.bounce_ball(self.ball)
        self.player2.bounce_ball(self.ball)

        # відбивання від верхньої й нижньої межі
        if (self.ball.y < self.y) or (self.ball.top > self.top):
            self.ball.velocity_y *= -1

        # вихід за межі поля й нарахування ігрових очок
        if self.ball.x < self.x:
            self.player2.score += 1
            self.serve_ball(vel=(4, 0))
        if self.ball.x > self.width:
            self.player1.score += 1
            self.serve_ball(vel=(-4, 0))

    def on_touch_move(self, touch):
        if touch.x < self.width / 3:
            self.player1.center_y = touch.y
        if touch.x > self.width - self.width / 3:
            self.player2.center_y = touch.y

class PongApp(App):
    def build(self):
        game = PongGame()
        game.serve_ball()
        Clock.schedule_interval(game.update, 1.0 / 60.0)
        return game

if __name__ == '__main__':
    PongApp().run()
```

Остаточна версія .kv файлу має вигляд:

```
#:kivy 1.0.9

<PongBall>:
    size: 50, 50
    canvas:
        Ellipse:
            pos: self.pos
            size: self.size

<PongPaddle>:
    size: 25, 200
    canvas:
        Rectangle:
            pos:self.pos
            size:self.size

<PongGame>:
    ball: pong_ball
    player1: player_left
    player2: player_right

    canvas:
        Rectangle:
            pos: self.center_x-5, 0
            size: 10, self.height

    Label:
        font_size: 70
        center_x: root.width / 4
        top: root.top - 50
        text: str(root.player1.score)

    Label:
        font_size: 70
        center_x: root.width * 3 / 4
        top: root.top - 50
        text: str(root.player2.score)

    PongBall:
        id: pong_ball
        center: self.parent.center

    PongPaddle:
        id: player_left
        x: root.x
        center_y: root.center_y

    PongPaddle:
        id: player_right
        x: root.width-self.width
        center_y: root.center_y
```

Запустіть проект і протестуйте його роботу.

4. Завдання для виконання

1. Прочитайте текст та виконайте завдання й приклади, що в ньому зустрічаються.
2. Вдоскональте ігрову графіку за своїм смаком. Наприклад, у багатьох графічних примітивів (прямокутники, еліпси й т.п.) є поле source, що містить ім'я графічного файлу, що використовується як текстури для заповнення об'єкту.
3. Реалізуйте умову виграшу (наприклад, коли гравець набирає 10 очок) і виводьте повідомлення про перемогу. Підказка: це можна зробити за допомогою `Label`, що з'являється при виконанні певної умови. Знайдіть в документації функції `add_widget` та `remove_widget`, що дозволяють динамічно додавати й видаляти віджети.
4. Створіть кнопки для старту, паузи й перезапуску гри. Підказка: скористайтеся віджетами `Button`.
5. Нині багато пристроїв мають підтримку Multi-Touch. Реалізуйте варіант гри для 4 ракеток (по одній на кожній межі поля).
6. Вдоскональте механіку зіткнення м'яча з ракеткою. Наприклад зробіть відбивання від країв ракетки більш реалістичним та/або враховуйте вплив швидкості ракетки на швидкість м'яча.

Використані джерела

1. https://en.wikipedia.org/wiki/Kivy
2. https://kivy.org/docs/tutorials/pong.html
3. https://kivy.org/docs/installation/installation.html