Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Лабораторная работа №3
«Управление жестами в Android»

Выполнил А.С. Бригадир

Проверил О.М. Внук

МИНСК 2024

1 ЦЕЛЬ РАБОТЫ

Целью работы является:

– определить не менее 2-3 сложных действий для управления интерфейсом приложения и добавить их в реализуемое мобильное приложение;

– продемонстрировать работоспособность жестов.

2 ИСХОДНЫЕ ДАННЫЕ К РАБОТЕ

Среда разработки Android Studio.

Язык программирования Kotlin.

Источник, содержащий исходный код: https://github.com/nankokit/tamagotchi_app.

3 ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ

Управление жестами является важной частью пользовательского интерфейса мобильных приложений, позволяя пользователям взаимодействовать с приложением более интуитивно. В Android разработчики могут использовать различные жесты, такие как касания, свайпы и многопальцевые взаимодействия, чтобы улучшить опыт использования.

Существует несколько основных типов жестов. Однопальцевые жесты включают тап — простое касание экрана, которое обычно используется для выбора элементов, и долгое нажатие, которое может открывать контекстные меню или выполнять дополнительные действия. Свайп, или движение пальцем по экрану, также широко используется для навигации между экранами приложения. Многопальцевые жесты, такие как сжатие и растяжение, позволяют пользователям масштабировать изображения или карты, а перетаскивание с несколькими пальцами дает возможность перемещать элементы интерфейса.

Для реализации жестов в Android существуют несколько подходов. Один из них — использование класса `GestureDetector`, который упрощает обработку различных жестов, таких как нажатия, свайпы и двойные касания. Также разработчики могут воспользоваться классом `ScaleGestureDetector`, предназначенным для обработки жестов масштабирования. Кроме того, можно использовать интерфейс `OnTouchListener` для обработки касаний на уровне `View`, что позволяет создавать более сложные жесты, комбинируя разные типы взаимодействий.

Примеры сложных действий для управления интерфейсом могут включать свайп для переключения между фрагментами приложения, где пользователь может провести пальцем влево или вправо для навигации. Другим примером является масштабирование изображений с помощью жеста сжатия, когда пользователь использует два пальца для изменения размера изображения. Также можно реализовать перетаскивание элементов в списке,

что позволяет пользователям изменять порядок задач или элементов.

Для демонстрации работы жестов можно создать простое приложение с несколькими функциями. Например, реализация `ViewPager` позволит пользователю переключаться между фрагментами с помощью свайпов. Использование `ImageView` и `ScaleGestureDetector` даст возможность масштабировать изображения, а `RecyclerView` с поддержкой перетаскивания позволит изменять порядок элементов списка.

Таким образом, управление жестами в Android значительно улучшает взаимодействие пользователя с приложением, делая его более интуитивным и удобным. Правильная реализация жестов может повысить удовлетворенность пользователей и улучшить общий опыт работы с приложением.

4 РАЗРАБОТКА ПРОЕКТА

Свайп — это горизонтальное движение пальца по экрану, которое может быть направлено влево или вправо. В зависимости от направления приложения выполняет определенные действия, например, переключается на следующий или предыдущий экран.

Реализация свайпа в Android-приложениях основана на понимании работы системы событий касания.

Система событий касания в Android генерирует события, такие как ACTION_DOWN (начало касания), ACTION_MOVE (движущееся касание) и ACTION_UP (окончание касания). Для обнаружения свайпа необходимо отслеживать смещение между начальным и конечным положением касания.

Определение направления свайпа происходит на основе расчета смещения. Если смещение больше нуля, это указывает на движение вправо, если меньше нуля – влево.

Хотя механизм обнаружения свайпа может быть одинаковым во всех активностях, действия, выполняемые после его обнаружения, могут сильно отличаться в зависимости от конкретной задачи активности. Например, в одной активности может происходить переход к другой странице, а в другой - изменение анимации.

При разработке системы обнаружения свайпа важно учитывать производительность, особенно на устройствах с ограниченными ресурсами. Это может включать ограничение частоты обновлений или применение различных алгоритмов для уменьшения нагрузки на процессор.

Реализация свайпа должна быть универсальной и работать корректно на различных устройствах и версиях Android, учитывая возможные различия в размерах экранов и разрешениях. Правильно реализованная система свайпа должна обеспечить интуитивно понятную навигацию для пользователей, позволяя им легко перемещаться между различными разделами приложения.

Важно также учитывать совместимость с общим дизайном приложения и не нарушать пользовательский опыт, созданный другими элементами интерфейса.

Правильная реализация свайпа требует комплексного подхода,

учитывающего технические аспекты работы Android, пользовательский опыт и специфику конкретного приложения. Это позволяет создавать интуитивно понятные и удобные интерфейсы, которые улучшают общую производительность и удовлетворенность пользователей.

В приложении реализованы свайпы между карточками питомцев на странице выбора питомца. Изображение этой страницы приведено на рисунке 4.1.

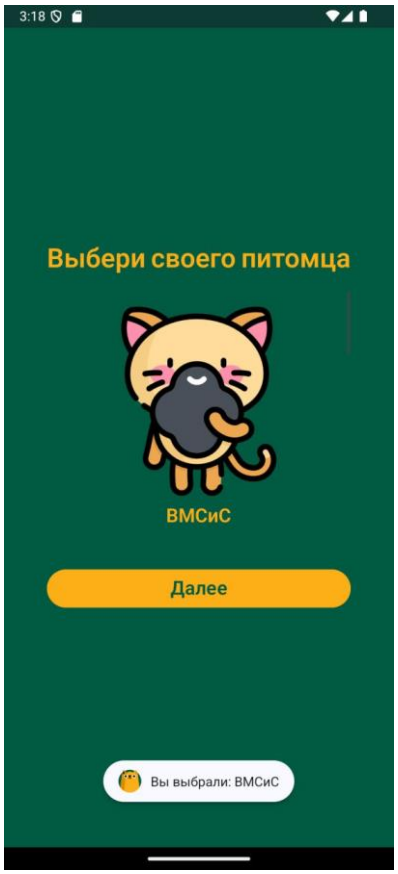

Рисунок 4.1 – Страница создания питомца

Перетаскивание (Drag and Drop) — это интерактивный жест, который позволяет пользователю перемещать объекты на экране, удерживая их и перетаскивая в другое место. Эта функция часто используется для улучшения взаимодействия с пользователем и позволяет создавать более интуитивно понятные интерфейсы. Основные компоненты Drag and Drop включают начало перетаскивания, перетаскивание и завершение перетаскивания. Перетаскивание начинается с события ACTION_DOWN, когда пользователь касается элемента, который хочет переместить. В этот момент необходимо создать объект ClipData, который будет содержать информацию о том, что именно перетаскивается. Во время перетаскивания генерируется событие ACTION_MOVE, которое отслеживает движение пальца по экрану. В это время обновляется позиция перетаскиваемого объекта, чтобы он следовал за движением пальца. Перетаскивание завершается событием ACTION_UP, когда пользователь отпускает палец. В это время необходимо определить, был

ли объект сброшен на допустимую область, и выполнить соответствующее действие (например, переместить объект или изменить его состояние).

Для реализации перетаскивания в Android используется класс View.OnDragListener для обработки событий, связанных с перетаскиванием. Первым шагом является инициализация Drag: при касании объекта создается ClipData и вызывается метод startDrag(). Затем необходимо обработать события Drag, установив OnDragListener на целевом элементе. Это позволяет обрабатывать события начала перетаскивания, когда объект перетаскивания вошел в целевую область, движения объекта перетаскивания, а также завершения перетаскивания.

Эта функция реализована на главной игровой странице для изображений лапши, кофе и ноутбука, которые при перетаскивании на питомца увеличивают значения потребности.

Рисунок 4.2 – Главная игровая страница

5 ВЫВОДЫ

В ходе выполнения лабораторной работы по управлению жестами в Android были достигнуты определенные результаты. Были выбраны и успешно реализованы сложные действия для управления интерфейсом приложения. Среди них — свайп для пролистывания списка питомцев и перетаскивание игровых элементов.