Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

УДК 004.424 https://doi.org/10.33619/2414-2948/128/10

АДАПТИВНАЯ ПЕРЕКОМПОНОВКА ИНТЕРФЕЙСА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ПРИ АКТИВАЦИИ СЕНСОРНОЙ КЛАВИАТУРЫ В СРЕДЕ РАЗРАБОТКИ UNITY

*©**Абляев М. Р.**, ORCID: 0009-0008-2920-9496, Крымский инженерно-педагогический университет им. Ф. Якубова, г. Симферополь, Россия, ablyaev.marlen@gmail.com*

ADAPTIVE REFRAMING OF THE MOBILE APPLICATION INTERFACE WHEN ACTIVATING THE TOUCH KEYBOARD IN THE UNITY DEVELOPMENT ENVIRONMENT

*©**Ablyaev M.**, ORCID: 0009-0008-2920-9496, Crimea Crimean Engineering and Pedagogical University named after F. Yakubov, Simferopol, Russia, ablyaev.marlen@gmail.com*

Аннотация. Рассматривается проблема адаптивной перекомпоновки пользовательского интерфейса мобильного приложения при изменении доступного экранного пространства, вызванном появлением сенсорной клавиатуры. Актуальность исследования обусловлена тем, что при вводе текста часть элементов интерфейса может оказаться перекрытой, что снижает удобство работы пользователя и нарушает функциональность приложения. Цель работы состоит в разработке и описании механизма автоматической адаптации интерфейса в среде Unity, обеспечивающего сохранение видимости и доступности элементов управления в условиях уменьшения рабочей области экрана. В качестве основы предложенного подхода используются встроенные средства Unity, позволяющие определять текущее состояние сенсорной клавиатуры и измерять занимаемую ею область экрана с помощью объектов Screen, Camera и TouchScreenKeyboard. На этой основе выполняется пересчет положения интерфейсных элементов, реализованных через Canvas и RectTransform, с изменением якорей и смещений в соответствии с новой высотой видимой области. Практическая реализация метода показана на примере компактного скрипта, который сохраняет исходное положение интерфейсного контейнера и при появлении клавиатуры смещает его вверх на величину, равную высоте занятой области, а после скрытия клавиатуры возвращает интерфейс в исходное состояние. Полученные результаты показывают, что предложенный способ обеспечивает динамическую адаптацию интерфейса без ручного перераспределения элементов, сохраняет доступность кнопок и полей ввода, а также повышает устойчивость приложения к изменению параметров отображаемого пространства. Разработанный механизм отличается совместимостью с различными устройствами, гибкостью настройки, возможностью применения в формах, чатах и других сценариях ввода, а также улучшает пользовательский опыт за счет автоматического предотвращения перекрытия элементов интерфейса сенсорной клавиатурой.

Abstract. This article examines the adaptive reflow of a mobile app's user interface when the available screen space changes due to the addition of a touch keyboard. The relevance of this study lies in the fact that some interface elements may become obscured during text input, reducing user experience and disrupting the app's functionality. The goal of this work is to develop and describe a mechanism for automatic interface adaptation in the Unity environment that preserves the visibility and accessibility of controls as the screen's working area decreases. The proposed approach is based on Unity's built-in tools, which allow one to determine the current state of the touch keyboard and

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

measure the screen area it occupies using the Screen, Camera, and TouchScreenKeyboard objects. This allows the position of interface elements implemented via Canvas and RectTransform to be recalculated, with anchors and offsets adjusted to match the new height of the visible area. A practical implementation of this method is demonstrated using a compact script that preserves the initial position of the interface container and, when the keyboard appears, shifts it upward by an amount equal to the height of the occupied area. After the keyboard is hidden, the interface returns to its original state. The results demonstrate that the proposed method enables dynamic interface adaptation without manually rearranging elements, maintains the accessibility of buttons and input fields, and improves the application's resilience to changes in display space parameters. The developed mechanism is compatible with various devices, is highly customizable, and can be used in forms, chats, and other input scenarios. It also improves the user experience by automatically preventing the touch keyboard from overlapping interface elements.

Ключевые слова: мобильное приложение, сенсорная клавиатура, адаптивный интерфейс.

Keywords: mobile application, touch keyboard, adaptive interface.

Одной из ключевых задач разработки мобильных приложений является обеспечение корректного отображения контента при изменении доступного экранного пространства. Наиболее частой причиной его изменения является активация сенсорной клавиатуры при вводе текста. В отсутствие адаптивной механики часть интерфейса может быть скрыта, что ухудшает пользовательский опыт и снижает функциональность приложения [1].

В связи с этим актуально исследование методов автоматической перекомпоновки интерфейса при изменении параметров экрана, с акцентом на реализацию в среде разработки Unity – одной из наиболее популярных систем разработки кроссплатформенных приложений.

В Unity доступное – экранное пространство можно динамически определять средствами встроенных классов, таких как Screen, Camera и TouchScreenKeyboard. При запуске приложения система фиксирует исходные размеры рабочей области – полный видимый экран, доступный для отображения элементов интерфейса. Эти параметры считываются и сохраняются в память приложения для дальнейшего сравнения при изменении состояния [2, 3].

Как только пользователь активирует поле ввода, операционная система вызывает сенсорную клавиатуру, что приводит к уменьшению полезной высоты экрана. В Unity этот процесс отслеживается через объект TouchScreenKeyboard, предоставляющий информацию о состоянии клавиатуры и параметрах области, которую она занимает. Элемент TouchScreenKeyboard.area возвращает прямоугольник, соответствующий занимаемому экранному пространству клавиатуры, и его высота становится ключевым параметром для перерасчета видимой области интерфейса [4].

После определения новых границ рабочей зоны выполняется перерасчет позиционирования интерфейсных элементов. В большинстве случаев интерфейс приложения реализован через иерархию объектов Canvas с компонентами RectTransform, которые содержат сведения о положении и размерах элементов относительно родительского контейнера. При изменении доступного пространства программа должна обновить эти координаты. Это достигается путем изменения значения якорей и смещений элементов, чтобы они оставались в пределах видимой зоны, сохранив при этом визуальные пропорции.

Для практической реализации предложенного подхода в Unity можно использовать компактный скрипт, который отслеживает появление сенсорной клавиатуры и изменяет

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

положение интерфейсного контейнера в зависимости от высоты занятой ею области. Такой механизм позволяет динамически сохранять видимость основных элементов управления и обеспечивает корректную адаптацию пользовательского интерфейса к изменению доступного экранного пространства.

```
using UnityEngine;
public class KeyboardAdaptiveLayout : MonoBehaviour
{
   [SerializeField] private RectTransform contentRoot;
   private Vector2 basePosition;
   private void Start()
   {
      if (contentRoot == null)
         contentRoot = GetComponent<RectTransform>();
      basePosition = contentRoot.anchoredPosition;
   }
   private void Update()
   {
      if (TouchScreenKeyboard.visible)
      {
         float keyboardHeight = TouchScreenKeyboard.area.height;
         contentRoot.anchoredPosition = basePosition + new Vector2(0, keyboardHeight);
      }
      else
      {
         contentRoot.anchoredPosition = basePosition;
      }
   }
}
```

Данный код фиксирует исходное положение интерфейсного контейнера и при появлении сенсорной клавиатуры смещает его вверх на величину, равную высоте клавиатуры. После скрытия клавиатуры интерфейс возвращается в первоначальное состояние. Такой подход обеспечивает сохранение видимости активных элементов интерфейса в пределах доступной рабочей области экрана [5].

В качестве примера, если приложение содержит форму ввода с кнопками подтверждения, размещенными в нижней части экрана, при появлении клавиатуры они будут перекрыты. Адаптивный алгоритм должен автоматически перемещать эти элементы вверх, учитывая высоту клавиатуры, либо изменять масштаб основного контейнера, чтобы уменьшить высоту контента и освободить место для клавиатуры. Таким образом, перераспределение происходит без потери части интерфейса и без сдвига элементов за пределы видимой области [3].

Для обеспечения плавности процесса используется непрерывная проверка состояния клавиатуры внутри игрового цикла (метод Update). Когда клавиатура становится видимой, запускается функция перерасчета пространства, где из исходной высоты экрана вычитается высота клавиатуры, и полученное значение передается в функцию изменения интерфейса. После деактивации клавиатуры все элементы возвращаются к исходным координатам, что гарантирует стабильность визуальной структуры приложения.

Т. 12. №7 2026
https://doi.org/10.33619/2414-2948/128

Следует подчеркнуть, что ключевым моментом реализации является взаимодействие между логикой приложения и адаптивной верстkой интерфейса. Разработчик должен обеспечить не только корректное перемещение элементов, но и сохранение их функциональных связей – например, кнопки должны оставаться доступными, поля ввода не должны терять фокус, а анимации интерфейса должны оставаться синхронизированными [2]. Для этого часто применяется буферная система хранения состояний интерфейса, которая позволяет быстро возвращаться к предыдущей компоновке без необходимости полного пересоздания всех элементов.

Предложенный метод адаптивной перекомпоновки имеет ряд преимуществ: совместимость (подходит для большинства устройств и операционных систем, поддерживаемых Unity); гибкость (позволяет использовать один интерфейс для портретного и ландшафтного режимов без дополнительной вёрстки); улучшение UX (обеспечивает удобный доступ ко всем функциям приложения, даже при активном вводе текста); автоматизация (снимает необходимость ручного позиционирования элементов при каждом изменении размеров экрана, снижая риск ошибок верстки); масштабируемость (легко интегрируется в существующую архитектуру приложения, позволяя адаптировать метод под различные типы интерфейсов – формы, чаты, поля ввода и т.д.).

Разработка механизма адаптивной перекомпоновки интерфейса при активации сенсорной клавиатуры является важным аспектом повышения качества и удобства мобильных приложений. Реализация данной функции в Unity позволяет динамически управлять расположением элементов и корректно отображать контент независимо от устройства и ошибок масштабирования. В результате достигается улучшение взаимодействия пользователя с приложением и повышение общей устойчивости интерфейса к изменению параметров отображаемого пространства.

Список литературы:

1. Gkoumas A., Komninos A., Garofalakis J. Usability of visibly adaptive smartphone keyboard layouts // Proceedings of the 20th Pan-Hellenic Conference on Informatics. 2016. P. 1-6. https://doi.org/10.1145/3003733.300374

2. Vertanen K., Kristensson P. O. Complementing text entry evaluations with a composition task // ACM Transactions on Computer-Human Interaction (TOCHI). 2014. V. 21. №2. P. 1-33. http://dx.doi.org/10.1145/2555691

3. Brown E., Buxton W., Murtagh K. Windows on tablets as a means of achieving virtual input devices // Interact. 1990. P. 675-681.

4. De Byl P. Holistic Mobile Game Development with Unity. Routledge, 2014.

5. Young C. J. Unity production: Capturing the everyday game maker market // Game production studies. 2021. P. 141.

References:

1. Gkoumas, A., Komninos, A., & Garofalakis, J. (2016, November). Usability of visibly adaptive smartphone keyboard layouts. In *Proceedings of the 20th Pan-Hellenic Conference on Informatics* (pp. 1-6). https://doi.org/10.1145/3003733.300374

2. Vertanen, K., & Kristensson, P. O. (2014). Complementing text entry evaluations with a composition task. *ACM Transactions on Computer-Human Interaction (TOCHI)*, *21*(2), 1-33. http://dx.doi.org/10.1145/2555691

3. Brown, E., Buxton, W., & Murtagh, K. (1990). Windows on tablets as a means of achieving virtual input devices. In *Interact* (pp. 675-681).

4. De Byl, P. (2014). *Holistic Mobile Game Development with Unity*. Routledge.

5. Young, C. J. (2021). Unity production: Capturing the everyday game maker market. *Game production studies*, 141.

Поступила в редакцию
05.05.2026 г.

Принята к публикации
11.05.2026 г.

Ссылка для цитирования:

Абляев М. Р. Адаптивная перекомпоновка интерфейса мобильного приложения при активации сенсорной клавиатуры в среде разработки Unity // Бюллетень науки и практики. 2026. Т. 12. №7. С. 96-100. https://doi.org/10.33619/2414-2948/128/10

Cite as (APA):

Ablyaev, M. (2026). Adaptive Reframing of the Mobile Application Interface when Activating the Touch Keyboard in the Unity Development Environment. *Bulletin of Science and Practice, 12*(7), 96-100. (in Russian). https://doi.org/10.33619/2414-2948/128/10