**DASHING THROUGH THE DESERT:**

**PLAYER EXPERIENCE AND MECHANICS**

*Prof. Reshma Lohar, Guide, Rizvi College of Engineering, Mumbai, India, reshmalohar@eng.rizvi.edu.in*

*Sagarika Srivastava, Game Developer, Rizvi College of Engineering Mumbai, India, sagarikasrivastava46@eng.rizvi.edu.in*

*Sidra Shaikh, Game Designer, Rizvi College of Engineering Mumbai, India,* [*sidrashaikh@eng.rizvi.edu.in*](mailto:sidrashaikh@eng.rizvi.edu.in)

*Bushra Ansari, Game Programmer, Rizvi College of Engineering Mumbai, India,* [*ansaribushra@eng.rizvi.edu.in*](mailto:ansaribushra@eng.rizvi.edu.in)

*Sidra Solkar, Game Animator, Rizvi College of Engineering Mumbai, India,* [*sidrasolkar2920@eng.rizvi.edu.in*](mailto:sidrasolkar2920@eng.rizvi.edu.in)

***Abstract:***

***This paper presents the development of "Desert Dino Dash," a 2D side-scrolling endless runner game. The game features a dinosaur character navigating through a dynamically generated environment filled with obstacles and collectables. The primary objective of the game is to score points by collecting in-game items and avoiding obstacles. The core gameplay mechanics was collision detection, physics, and level generation are discussed. We also the game design principles employed, including visual aesthetics and user engagement strategies. Moreover, the challenges faced during development, such as optimization for mobile devices and balancing gameplay difficulty, are explored. Future scope for improvement and the implementation of additional features is also addressed.***

***Keywords:***

## *2D game, Desert Dino Dash, endless runner, game mechanics, mobile game*

## INTRODUCTION

*Desert Dino Dash* is a 2D side-scrolling game developed using Unity, where players guide a dinosaur through a treacherous desert landscape filled with obstacles. The objective is to help the dinosaur safely reach its home while avoiding hazards like cacti, sandstorms, and other natural obstacles. The game incorporates elements of procedural generation to create varying levels of difficulty and increase replay ability.

This project was undertaken to explore key aspects of game development, including character control, physics interactions, and procedural content generation, using the Unity game engine. Unity's powerful 2D toolset, combined with C# scripting, enabled the creation of smooth animations, dynamic environments, and an engaging player experience.

The development process also included the use of design patterns and performance optimization techniques to ensure that the game runs efficiently across different platforms are discussed in detail, providing insight into how the game was structured and developed.

# LITERATURE SURVEY

## 

The development of *Desert Dino Dash* is built upon a foundation of existing work in the fields of game design, procedural generation, and 2D game development using Unity.

## *A. Unity Game Engine*

Unity is a widely used game engine that supports both 2D and 3D game development. Its versatility and extensive documentation make it an ideal platform for indie developers and professionals alike. The official Unity User Manual provides comprehensive guidelines for utilizing the engine’s 2D toolset, which was instrumental in the development of *Desert Dino Dash*. Unity's physics engine, rendering pipeline, and animation tools were all central to crafting the interactive environment and player experience. Goldstone’s *Unity 2021 Cookbook* also offered valuable insights into advanced features, such as optimization techniques and working with the 2D tile map system.

*B. Procedural Generation in Games*

Procedural generation plays a significant role in creating dynamic and engaging gameplay by generating new content on the fly. The use of procedural algorithms in game design was found out to be very popular. Adams' *Game Mechanics: Advanced Game Design* highlights the importance of creating diverse and unpredictable environments, a concept that directly influenced the design of the cactus spawner system in *Desert Dino Dash*. The procedural generation of obstacles adds replay ability by ensuring that no two levels are exactly the same, providing players with a unique challenge each time they play.

## *C. Game Design Patterns*

The application of game design patterns is crucial for structuring game logic so that it promotes flexibility and maintainability. Nystrom’s *Game Programming Patterns* offers in-depth explanations of common patterns like the "State" and "Observer" pattern, which are widely used in Unity projects. These patterns were leveraged in *Desert Dino Dash* to manage the game’s state transitions, handle user input, and control character behaviour. The use of such patterns ensures that the code is modular, easy to debug, and scalable as new features are introduced.

*D. 2D Animation and Character Control*

The development of smooth and responsive character animations is essential for player immersion. Unity’s Animator system allows for the creation of complex animation state machines, which was pivotal in controlling the dinosaur’s movements, from running to jumping. Goldstone's *Unity 2021 Game Development Essentials* details the use of the Animator system and helped guide the setup of character animations in *Desert Dino Dash*. Additionally, Stephens’ *C# Programming Cookbook* provided insights into scripting the character’s movement and collision detection, ensuring that the character reacts naturally to obstacles like cacti.

*E. Asset and Environment Design*

Designing a visually appealing yet optimized game environment is a key challenge in game development. Callow’s work on virtual reality projects, while focused on 3D assets, provided valuable strategies for managing performance through efficient asset usage and optimization techniques, which were adapted to suit the 2D environment of *Desert Dino Dash*. The desert-themed environment in game is a procedurally generated asset used to create a dynamic and visually engaging atmosphere.

*F. Sound and Music in Games*

Sound design is an underappreciated aspect of game development, it plays a significant role in enhancing the player experience. Integrating sound effects, like the dinosaur’s footsteps and the rustling of the desert wind, required understanding how to trigger audio events based on in-game actions. Unity Asset Store and Unity's official

**3. EXPERIMENTAL SETUP AND METHODOLOGY**

The development of *Desert Dino Dash* was conducted using Unity, a widely used game engine for 2D and 3D game development. The project was developed on a system with an Intel Core i7-9700K CPU, 16 GB of RAM, and an NVIDIA GeForce GTX 1660 GPU, running Windows 10. For coding, Visual Studio 2022 was used as the integrated development environment (IDE), with C# as the primary scripting language. Version control was managed using Git and GitHub, ensuring smooth collaboration and backup of project files. Graphics for the game, such as character sprites and environmental assets, were created using Adobe Photoshop and GIMP, while Audacity was employed for sound editing.

The game design process started with the conceptualization of a side-scrolling desert environment where players control a dinosaur navigating through obstacles. Early prototypes were built to test core mechanics, including character movement and physics interactions with environmental hazards like cacti. These prototypes allowed for iterative design, where gameplay features were continuously refined based on feedback from playtesting sessions. Play testers provided insights into the game’s difficulty, responsiveness, and overall enjoyment, which guided subsequent development stages.

A significant aspect of the game was the implementation of procedural generation for obstacle placement. A cactus spawner script was developed to randomly generate obstacles at varying intervals and positions along the game's landscape. This was achieved using Unity’s Random. Range function, ensuring unpredictability while maintaining fairness in gameplay by avoiding overlapping obstacles. As the player progresses, the game’s difficulty increases by reducing the time interval between spawns, creating a more challenging environment. Collision detection was also integrated to ensure accurate interaction between the dinosaur and the obstacles.

The character’s movement and control were implemented using Unity’s Rigidbody2D component, allowing for realistic physics-based movement, including running and jumping. Animations for the character were handled using Unity's Animator system, which managed smooth transitions between different states like running, jumping, and idle based on player inputs. This setup ensured that the characterresponded fluidly to controls while maintaining visual consistency.

The user interface (UI) was designed with simplicity in mind, consisting of essential elements like the score display and a start/restart menu. The score system, displayed via a TextMeshPro component, tracked the player’s progress throughout the game. Menus were built using Unity’s UI system, with buttons triggering specific game states such as starting the game or resetting after game-over event.

Throughout development, a combination of unit testing and playtesting ensured the game’s functionality and smooth performance. Unit testing focused on individual components like the cactus spawner and character movement, while playtesting involved external users providing feedback on the gameplay experience. Performance optimization was carried out using Unity’s Profiler, which helped identify and resolve issues to ensure the game ran efficiently across different devices. Asset compression and minimizing the number of active game objects were among the techniques used to optimize performance.

### 4. CONCLUSION

The development of *Desert Dino Dash* provided valuable insights into the game development process, from conceptualization to implementation using the Unity game engine. Through the integration of procedural generation, character animations, and responsive gameplay mechanics, an engaging-challenging experience was generated for players. The literature and resources surveyed in this journal, including Unity documentation and game design references, were pivotal in shaping the game’s development and addressing technical challenges.

This project not only enhanced understanding of game design patterns, scripting in C#, and asset optimization but also underscored the importance of creating a cohesive player experience through the combination of visuals, sound, and mechanics. The procedural nature of the game, particularly the cactus spawner, adds replay ability and randomness, ensuring that each playthrough presents new challenges. The utilization of Unity's physics and animation systems contributed to the smooth player interactions and visually dynamic environment.

In summary, *Desert Dino Dash* exemplifies how Unity's versatile 2D tools can be used to create a fun and engaging game, while also providing a platform for further exploration of game mechanics and procedural design. Future work could involve expanding the game’s features, including additional levels, new mechanics, and enhancing performance across different platforms

**REFERENCES**

1. M. A. Jones, 2020, Learning Unity 2019: A Hands-On Guide to Game Development, Pack Publishing, pp. 78–90.
2. K. E. Rodriguez and S. M. Adams, “An Evaluation of Unity as a Game Development Platform,” IEEE Access, vol. 8, pp. 12345–12355, 2020.
3. P. M. Green, “Game Prototyping with Unity: Best Praces,” in Proc. IEEE Int. Conf. on Games and Virtual Worlds for Serious Applications, 2019, pp. 110–116.
4. R. Adams, *Game Mechanics: Advanced Game Design*, New Riders Games, 2013.
5. W. Goldstone, *Unity 2021 Game Development Essentials*, Packt Publishing, 2021.
6. R. Nystrom, *Game Programming Patterns*, Genever Benning, 2014.
7. M. Callow, *Unity 2020 Virtual Reality Projects: Learn VR Development by Building Immersive Applications and Games*, 3rd ed. Packt Publishing, 2020.
8. Unity Technologies, *Unity User Manual (2023.3)*, Unity Technologies, 2023. [Online].
9. A. Goldstone, *Unity 2021 Cookbook: Over 100 Recipes to Take Your 2D and 3D Game Development to the Next Level*, 4th ed. Packt Publishing, 2021.