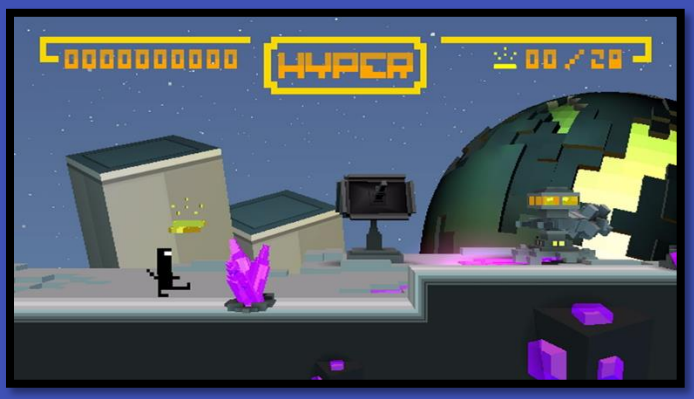
ENDLESS RUNNER – FEEDBACK



PRACTITIONER LEVELS	
Talented	
Skilled	
Accomplished	
Capable	
Ineffectual	

GRADING	ID: 16025481
GAMEPLAY EXPERIENCE	
- HIGH SCORE SUPPORT	
TECHNICAL IMPLEMENTATION	
- PROCEDURAL LEVEL GENERATION	
GIT-FLOW MODEL	
	PASS

COMMENTS

Well done; this is a solid implementation of the game. The graphics are nice and consistent, there was an issue with font loading which was discussed. The scoreboard could be improved with names. The animation feels a little janky, but overall a solid and very playable game. As discussed there is plenty of scope for tweaks i.e. the collision handling on blocks. I particularly loved the way you introduced the game and its control scheme. Very well done.

Be careful with memory, just ensure it doesn't leak if the texture fails better yet use unique pointers. Try to use constants for things like the game width instead of 1280. Going forward a proper game state system would help reduce the complexity and roles of the main game class. A very strong codebase, with some room for improvement.

PREVIOUS ACTION POINTS
1. Improve on OOP practices
2. Be more considerate with use of dynamic memory
3.

ACTION POINTS
1. Use of smart pointers
2. State management system
3.

GENERAL FEEDBACK

GAMEPLAY	
101	Well done on creating an enjoyable endless runner game
102	The game plays well but requires more innovation or depth
103	The procedural generation is good enough to provide replay value
104	The high score system has been both well designed and implemented
105	The game is buggy and/or difficult to play
106	The collision system works well, and obstacles can be avoided
107	The menu system does not work
108	The progression or scoring system has not been implemented
109	The high score system is poorly implemented or non-functioning
110	The game or the UI lacks polish
111	Changes to the mechanics have subverted the game too aggressively
112	Lacks gameplay testing and/or balancing
113	Collisions are not working correctly

TECHNICAL	
201	You should be pleased, there's lots of solid design and code decisions
202	Functions are short and designed well to reduce code complexity
203	OOP has been used well throughout the game's design
204	There is good usage of composition
205	The coding standard has been adhered to
206	Const correctness has been used for both functions and objects
207	The STL has been used effectively
208	Top use of dynamic memory and in particular smart pointers
209	The codebase is reasonably well structured, but could be improved
210	Data is not being properly encapsulated
211	Memory has been misused and/or there are leaks
212	Functions and conditionals could be made more succinct
213	Make use of keywords default and delete
214	The procedural generation code is complex and hard to follow
215	Inherited classes do not have virtual destructors or use override
216	There is a general need to improve OOP design
217	The coding standard has been misused
218	Make sure to pass larger data structures by reference
219	The STL containers usage is not correct or non-optimal
220	The state system responsible for managing the game needs work
221	You have not used the git-flow process correctly