University of the Witwatersrand, Johannesburg
School of Electrical and Information Engineering
Software Development II

PROJECT ASSESSMENT FORM v0.34

Mwaniki 87%

Discretionary Mark: 5

Sethosa 87%

Discretionary Mark: 5

Bonus + Penalty Total: 0

Bonus, penalty and discretionary marks have *already been applied* to the marks shown.

	Unacceptable	Poor	Acceptable	Good	Excellent
Problem Understanding, Solution and Evaluation: project report	extremely flawed problem understanding/specification/ conceptual solution, key functionality and/or design choices not explained, presentation of solution does not match implementation	poor problem understanding/ specification/conceptual solution, key functionality and/or design choices hardly explained, class responsibilities inadequately described, blind acceptance of clearly defective functionality	adequate problem understanding/specification and conceptual model, class responsibilities described, some description of dynamic behaviour, minimal critique of the final solution in terms of functionality and design	good understanding/specification and conceptual model, class responsibilities well described, reasonable description of dynamic behaviour, reasonable critique of the final solution in terms of both functionality and design	astute understanding, specification and conceptual model, class responsibilities and dynamic behaviour are well described, excellent critique of the final solution in terms of both functionality and design, consideration of the broader problem domain
C++ Design and Implementation: source code	SFML 2.5.0 not used, implementation violates constraints, not object-oriented: no user-defined classes, GitHub not used for version control	poorly chosen abstractions or many key abstractions missing, poorly designed class interfaces, inappropriate relationships between classes, patent violation of fundamental principles such as DRY, implementation more like C than C++	abstractions generally have acceptable/appropriate behaviour but some key ones may be missing, acceptable class interface design but implementation may not be well hidden, mostly acceptable class relationships, modern, idiomatic C++17 mostly used	*2 out of 4* 1) well-modelled abstractions at all levels of granularity with good interfaces which hide information 2) clean separation of presentation and logic layers 3) small classes and no long functions 4) good use of role modelling. No clearly wrong design decisions, modern, idiomatic C++17 used	*3 out of 4*: 1) well-modelled abstractions at all levels of granularity with good interfaces which hide information 2) clean separation of presentation and logic layers 3) small, cohesive classes and no long functions 4) good use of role modelling. No clearly wrong design decisions, modern, idiomatic C++17 used
Functionality: game executable	no executable, executable does not run	application has major functional flaws, no splashscreen with playing instructions	all basic functionality working acceptably, splashscreen with playing instructions	all basic functionality working plus 3 minor features (one of which must be the mushroom field) OR 1 major feature and 1 minor feature (the mushroom field), splashscreen with playing instructions	all basic functionality working plus 2 major features (one of which must be independently-moving centipedes) and 2 minor features, splashscreen with playing instructions
Automated Testing: test executable and source code, test section in project report	no genuine attempt at unit testing, doctest framework not used	a small proportion of functionality is tested, testing inadequate	test coverage of game logic is adequate and includes basic movement and collision testing for all game objects, some important game logic is not tested, adequate test section in report	test coverage of game logic includes all classes/functions responsible for the movement and collision of game objects, either directly *or indirectly*, testing is thorough and test code is of good quality, good test section in report	distinguished from *Good* by one or more factors comprehensive coverage of all game logic, advanced use of testing framework, use of a mocking framework, automated tests given for difficult-to-test functionality eg. involving randomness, gui interactions etc
Technical Communication: project report and technical reference manual	report deviates significantly from the School's standards, technical reference manual not generated using Doxygen	report does not conform to the school's standards, use of language, style and tone is poor, report structure is poor, poor technical reference manual	report mostly conforms to the school's standards, use of language, style and tone is acceptable, report structure is acceptable, acceptable technical reference manual	report mostly conforms to the school's standards, use of language, style and tone is good, good use of diagrams to communicate concepts, report is well-structured, **acceptable abstract**, good technical reference manual	report fully conforms to the school's standards, use of language, style and tone is excellent, good use of diagrams to communicate concepts, report is well-structured, **good abstract** and technical reference manual
Comments:	**This game is incredibly well done with both scorpions and spiders in addition to centipede splitting. The sound effects really add to the feel of the game - excellent!** **The depth and comprehensiveness of the test code is outstanding!** Try to avoid looping and branching in tests, and complex test code. Overall the object-oriented modelling has been done extremely well. There are many fine, cohesive abstractions including SpriteSheet, SeperatingAxisTheorem, SpatialHash, MushroomFactory, and so o A sophisticated approach has been taken in modelling and implementing the system. Role modelling is used with IEntity and IMovingEntity but this results in repeated code for many of the virtual functions, provide default implementations in the base class. EnemyFactory, GameEngine, Logic have many, disparate responsibilities. The report abstract is poor with no specific details pertaining to your excellent solution. Otherwise the report is well written with generally good diagrams (Figure 7 and 8 are not readable). The conceptual model is explained very well with accompanying diagrams. It would have been good to use a diagram for explaining the SAT. A detailed sequence diagram is given for run-time interactions. This could have been split into a number of smaller diagrams. The critiques are sound and some astute observations are made with respect functionality and with respect to the design and the missing Centipede Train class, in particular. **Overall excellent!**				

Notes:
All categories are equally weighted
If any category receives a rating of Unacceptable then both students' marks are capped at 40%
If any rating falls within a shaded block then the result will be a FCOM for both students for the course

Bonus and Penalties:
Non-compliant submissions: first: -5; second: -5; final: -5
Early hand-in: +5; Late final submission: within first hour: -5; before 16h30: -15