Risk Assessment – ‘Tether’ Game

**Gameplay experience:**

* Physics: the game is almost completely reliant upon predictable, replicable physics interactions. This will require a large amount of time allocated to programming, testing and amending values to ensure reliability.
  + Team must ensure that sufficient playtesting time is allocated in project timeline to allow for subsequent development based on player feedback and gameplay observations.
* The game depends upon the players being able to recognise game obstacles, cooperatively organise themselves in response to obstacles, and allocate tasks to overcome obstacles in a rapidly repeating cycle.
  + Telegraphing which obstacle is currently active and requires player attention will be crucial. The team has ideas to make this clear to the player, but this will require playtesting to confirm whether successful.
  + Game balance must be weighted/timed well. Playtesting will be the basis for this evaluation. The team is aware that if this is not polished significantly, the game will either be too easy or impossible to complete (or an unpredictable middle ground) which will put players off from continuing to play.
* Level design: through Prototyping it has become apparent that the static-environment the team had envisaged does not yield gameplay that is engaging over replays.
  + Designing levels, scenarios and puzzles which the players must navigate through together would overcome this issue, though this would mean the design of level layouts, various puzzles/hazards and environments which the team has no experience in and very limited time in which to produce something of good quality.
  + An alternate approach may be to keep the initial screen-spaced environment and generate obstacles which approach the player which must be avoided – staying on-screen to win. This would require level design work but limit the number of unique layouts and models required.
  + Playtesting will be crucial to design viable levels, team must account for this in project timeline.

**Visual assets:**

* art assets: while team has limited experience with 3D modelling software, the quality of assets produced would likely not be high. Team would also be inefficient at producing assets.
  + To mitigate this, team would either look to purchase assets/use free assets or outsource work from others. This does raise further potential issues with reliability/availability of artists and whether they can work to our projects schedule. Sourcing free assets/paying for assets also restricts team to what is available on the market, also meaning some assets may not continue the overall design choice throughout the project (e.g. cartoon style vs realistic).
* Animations: animations have the potential to be an effective tool in telegraphing the active obstacle to the player.
  + Team has little to no experience with this. Will either have to outsource work or devote time to learning how to complete the task themselves.
  + Potential solution (though not as effective) would be to use particle effects within the Unity engine to denote importance. Team has experience with this approach, and although the desired effect is rarely a quick process, the team are confident they can produce necessary quality of work.
  + Another viable solution is to use animation graph within unity to enact simple animations – allowing items to bounce, stretch or scale to draw attention. This is fairly quick to include within a project and will be eye-catching, though much less effective than a custom, fluid animation.

**Programming:**

* 2 team members: neither has worked in a team of two programmers before. Be aware that over scoping could be a very real issue. Any art assets produced/edited will no doubt take vastly longer than usual group estimates.
  + Code and logic issues may prove easier to overcome with two programmers.
  + From having worked with multiple programmers as part of a larger team, there are likely to be issues with code compatibility between scripts/issues understanding each other’s code if working on a script you did not begin yourself. Potential solution to limit confusion and wasted work time is to plan approaches before beginning a task – using class diagrams or agreeing on overall functionality of classes, which variables should be exposed, naming conventions and which scripts will need to interact with one another or be ‘manager’ scripts.
  + Though should be given to potential future additions/changes. To avoid extra work in the future and wasting of time, team should discuss which variables and how many are used to effect changes, and whether these will also be part of future algorithms.
* Physics: while unity does offer some niche tools to aid with physics simulation, they are also restrictive.
  + To overcome this issue, team can code their own physics behaviour to tailor the gameplay to the desired specifics.
  + A downside to this fix however, is the complexity of the task – taking a significant amount of time to produce, and further time to playtest for reliability. Team must account for this playtesting in the project timeline if this option is pursued.
* Profile upgrades/progress: If the final version has multiple levels, the game must be save-able so that players can exit the game and return without losing overall progress. User profiles may prove complex as team members have limited experience with this element on a small scale. From a quick search and brainstorm team members do believe it is well within team capability.
* Unity Engine: both members have experience developing for this engine – expect no unforeseen issues caused by this software.
* PC / Console platform: both members have experience developing for PC. Through the projects so far, both have learnt how to make games compatible for Xbox controller input. Porting game to console may well be beyond the scope of this year’s project.