The use of gamification within coastal management public consultation

Benjamin Mottram

March 31, 2014

Contents

1 Introduction

1.1 Abbreviations

SME: Subject Matter Expert.
ACMD: Advisory council of the misuse of drugs.
GM: Genetically Modified.
EAU: East Anglia University.
CRU: Climate Research Unit.
IPCC: The Independent Climate Change E-mail Review.

BGS: British Geological Survey.
FCERM: National Flood and Coastal Erosion Risk Management Strategy.
DEFRA: Department for Environment Food and Rural Affairs.
HTL, ATL, MR, NAI: Hold The Line, Advance The Line, Managed Realignment and No Active Intervention.
SMP: Shoreline Management Plan.
SAC, SSSI, AONB, SPA: Special Areas of Converstaion, Site of Special Scientific Intrerest, Area of Outstanding Natural Beauty, Special Protected Area.
RSPB: Royal Soceity for the Protection of Birds.
MA: Megaannum (1 million years).
NASA: North American Space Agency.
USGS: United States Geological Survey.
ETM: Enhanced Thermal mapper.
LEO: Low Earth Orbit.
GIS: Geographical Information Systems.
DARPA: Defense Advanced Research Projects Agency.
SDI: Spatial Data Infrastructure.
KML: Keyhole Markup Language.
OGC: Open Geospatial Consortium.
MMO: Massively Multiplayer Online Game.
DKP: Dragon Kill Points.
DSDM: Dynamic Systems Development Method.
MoSCoW: Must have, should have, could have, would like to have.
BVC: Big Visible Chart.

1.2 Overview

Coastaldefender is a project to create a game-like tool that will be designed to spark discussion around the specific topic of coastal defence management and flooding in the UK. This is particularly pertinent when considering the aftermath of the storms that descended upon the UK in the early months of 2014. The intensity of these storms could be seen to validate the decision to change the change in policy for coastal management and flood protection by DEFRA and the Environment Agency. Both institutions through the overarching FCERM are reacting to the mounting pressures of climate change to the UK's coastline where there once was blanket policy of holding the line,

new policies such as managed realignment where the coastline is re profiled are being considered.

The FCERM defers to Coastal Groups to decide policy for any given stretch of the coastline, they publish the Shoreline Management Plan to outline their intentions. However in some places (such of the Scolthead area of this study) the SMP clearly states more research is required before an optimal solution can be presented. Where research is lacking Coastaldefender would seek to crowd source information by providing a tool to generate interests and discussion within the subject. It is hoped that from these people it will interact with that some will develop ideas and strategies that could be employed. This concept of crowd sourcing has seen tremendous success in the past, and in the case of Goldcorp discussed in section 2.4.3 has turned an ailing business into one of the most formidable mining companies in the world.

The chosen technological platform for this project will be an Earth browser such as Google Maps or Bing Maps. The reasoning for this is that Earth browsers so ubiquitous within the internet, that any user can intrinsically understand it. Plus it also grounds the issue to a specific region, it provides context to the user as well as demonstrating the real world situation.

1.3 Project Aims

The project will seek to address the three following questions.

1. To determine if an Earth browser technology is subtitle for developing a game like tool.

2. To determine if a game like tool is good for developing discussion around a specific environmental or scientific issue.

3. To determine if there is a novel or specific crowd sourcing model that can be implemented to achieve the second aim.

1.4 Target Demographics

There is no target demographic for Coastaldefender. In previously successful crowd sourcing projects much of the value is found in the diversity of individuals they interact with. Since this project also will seek to integrate some

social media aspects, the social media platform it links to will have to be broad and far reaching.

1.5 Potential Limitations and Obstacles

Any project with this scope will have potential challanges facing it. It can be determined that the main challanges facing Coastaldefender will be.

- **Technology suitability:** Coastaldefener sits outside the typical usage for Earth Browser technology. Will there be any physical limitations on what can be achieved through the chosen API? Will it simply produce a product too cumbersome to use?
- **Crowd Sourcing Issues:** Where will the participants for this project be found from? Using exciting social network connections may make a representative slice of the populace but not necessarily an engaged one?
- **Context suitability:** Will users see the use of a game to promote a subject as something simply too frivolous for a serious discussion? Are there previous examples that set a precedent for this kind of project?

2 Literature Review

2.1 Overview

In 1985 the Royal Society published the Bodmer report [87], it was a response from what was seen as an increasing lack of engagement of the scientific community with the public. The report asserts many times over that it should be part of the scientist's job to communicate their specialist knowledge to the public. To explain, guide and describe.

When this communication fails, the so called public deficit model kicks in. A pubic with little scientific understanding becomes distrustful of technological advancements, and in some cases will physically object to it. Such was the case of vandalised GM crops in the early 2000s, with the vandals being acquitted by a court of peers who saw their actions having a moral imperative [53]. More extreme still are the ideologies that have emerged from an anti-technological stance. starting with Rousseau's idealised "noble

savage" [79, 82] they have evolved into the Peasentism championed by the Khmer Rouge [49], or the anarcho-primitivist beliefs of Ted Kaczynski who's campaign of bombing centres of technical excellence in the early 90s earned him the title of the Unibomber [51].

Even with the advent of the internet, and the potential outreach possible by scientists disinformation is pervasive. Conspiracy theories are rife and bizarre long dead scientific hypothesis like the expanding Earth have been resurrected on Youtube [1]. An example of how far reaching the implications of disinformation can be one can refer to the court case of The Kitzmiller vs. Dover Area School District [16], where members of the Dover area school board attempted to introduce biology text books that supported Biblical creationism under the guise of "intelligent design". The case eventually was found in favour of the plaintiffs, saying that it was indeed an attempt to circumnavigate the first amendment of the US constitution [45, 45, 16, 28, 54]. Interestingly enough it was the testimonials of SMEs in the field of biology that provided the bulk of the evidence for the plaintiff's case. It shows that when faced in debate a SME will trump proponents of psudo-science due to their knowledge and training.

Public engagement is therefore not simply about presenting the facts, since theses are often misinterpreted and misrepresented, hence an increasing desire to move engagement "up-stream", where the scientific community actively engages the public [91, 77, 48]. Some of these more successful attempts use crowd sourcing and game mechanics to engage online communities, Galaxay Zoo for example uses crowd sourcing to sort images of galaxies to tremendous effet [108], whilst Foldit a game based on folding proteins has used its incredibly competitive online community to solve a number of seemingly eleulsive problems [103].

This novel approach could be considered for the UK's coastal management challenges, increasing environmental pressure from climate change is forcing a strategy rethink in policy makers, but this is all predicated upon the correct research being available [10]. In some cases, such as the Scolthead region of North Norfolk the SMP policy documents state further research is needed before a more informed coastal management decision can be made [97]. considering this dearth of information this project proposes that using game mechanics within tools to reach out to existing online communities might not only be an effective way of engaging with the public about coastal management issues, but also as a tool for crowd sourcing possible alternative coastal management solutions.

2.2 Coastal Management

The UK is the quintessential island nation, and according to The British Cartographic Society Great Britain has a total coastline length of approximately 19,491 miles [85]. All of which is battered daily by the combined might of the North sea, the Irish sea and the Atlantic ocean; it is no surprise then that in some parts of the UK coastal erosion happens in in extreme rates. In one of the most extreme examples Happisburgh, Norfolk has seen 102 meters of land reclaimed by the sea between the period of 1992 and 2004, which is in excess of 8 meters a year.

According to the BGS 2012 geohazrd summary of coastal erosion [86], the hard metrics of coastal erosion's effect on the UK economy is the following.

- 113, 000 residential properties at risk via coastal erosion.
- 9000 commercial properties are at risk.
- 500 hectares of agricultural land is at risk.

This equates to £7 billion in assets currently at risk of loss by coastal erosion .

2.2.1 Coastal Management Oversight

According to official government policy all current decisions regarding coastal erosion falls under its FCERM. Currently the responsibility for resisting coastal erosion and coastal flooding is held by local authorities and the Environment agency, which amalgamate to form coastal erosion risk management authorities. These authorities in addition to their roles of erecting and maintaining coastal defences, have powers to restrict third party activity on any stretch of the coastline as well as removing beach material when required [98].

Actual policy decisions are made by Coastal groups, again these are amalgamations of various parties, and tend to consist of such actors as The Environment Agency, local authorities and other stakeholders such as Network Rail, English Heritage and the Natural Trust. It is from these groups that the Shoreline Management Plans are produced, which detail coastal defence in region by region basis [2].

As it stands the Flood Risk Management plan sees funding in the region of £800 million per annum, coastal erosion protection falls under this auspice.

The majority of this comes from DEFRA (£765 million), with contributions from the Scottish Parliament (£50 million) and the European Development Fund in Wales (£50 million). This is a budget increase authorised after the 2007 England and Wales floods which inflicted damages in excess of £3.2 billion, this also reflects a shift in policy to combat environmental issues exacerbated by climate change [10]. The current scope of the FCERM is to provide a national strategy for period between 2011 to 2030 [2].

2.2.2 The Shoreline Management Plan

The SMP is a collection of high level policy documents commissioned by the Coastal Groups. They provide an "intent of management" for the UK coastline(excluding Scotland).

Scope

There are 22 coastal groups in all, each consisting of a conglomeration of local authorities and stakeholders. Numbered sequentially as you travel along the UK coast with the first are being the mouth of the River Tyne and the last being Great Ormes Head.

Each are is defined as a "super frontage", a region that is subject to sedimentation processes local to itself. In the first round of SMPs the naming scheme for these super frontages reflected their position in a larger sediment cell (a sediment cell being an section of coastline under the influence of of sedimentation processes beholden to a specific source e.g. the Thames). Originally the super frontage this study is focusing on was frontage 3a. It has now been reclassified to SMP 5.

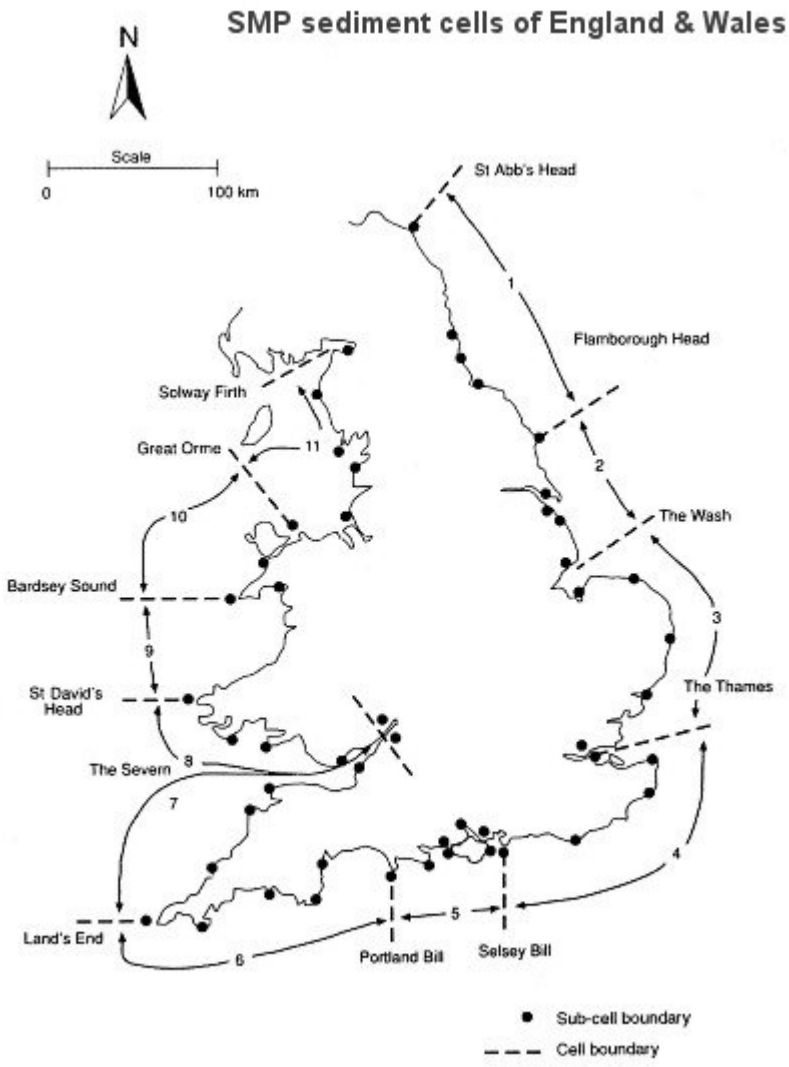

Figure 1: UK sediment cell systems.

The precise location of each super frontage was derived from the work of the Futurecoast study, and an internal natural England study undertaken by DEFRA and Halcrow in 2002. Both studies informed much of what is now the SMP layout and process.

The SMPs are currently in their second generation, with the first round published in 2001-2002. It is assumed that the poilcy will be reassessed every

ten years, with the third round of SMPs to appear in roughly 2021-2022. The benefit of this is to provide a policy that adapts to geological processes which normally slow, are highly changeable and occasionally catastrophic in the short term. This acceptance of dealing with long time frames is evident within their scope with policy for the short, medium and long term.

- Epoch 1: Is the present to 2025.
- Epoch 2: is 2026 to 2055.
- Epoch 3: is 2056 to 2105.

This long term planning allows for the Coastal groups to consider sustainability, which considering the impacts of climate change is an extremely prudent measure.

Policy Designations

Each coastal group draws its decision making from existing research of UK coast at both regional and national levels. However it is sometimes the case that extensive research is lacking and decisions have to be made without this additional input. This is the case in more than one section that this study covers. All policy designations are considered within ten guiding principles, which are stated verbatim from the document.

1. To manage the coast to reduce reliance on defences and to promote flexible coastal management options for present and future generations.
2. To ensure that local policy decisions do not adversely affect wider natural coastal processes.
3. Work with coastal change to take account of uncertainty about the future in the timing of policies.
4. To consider social and economic well being and allow communities and individuals to adapt to coastal change.
5. To consider the effects of coastal change on local industries (tourism, agriculture, fisheries, etc).
6. To take account of the value of the North Norfolk coast area to wider society.

7. To ensure that the timing of the policies allows the land use planning system to respond to any shoreline management changes and their consequences.

8. To contribute to maintaining and enhancing protected sites and species, subject to natural change.

9. To support maintenance and enhancement of biodiversity in the wider coastal countryside.

10. To contribute to maintaining and enhancing the character of the coastal landscape.

11. To have regard for the historic environment and its value for the heritage, culture and economy of the area.

Of course many of these guidelines are contradictory when making decisions in the real world, so both the FCERM and the SMPs stress pragmatism when required.

Policy is outlined with deceptive simplicity with any given stretch of the coastline being designated in one of four ways.

- **Hold the line (HtL)** this involves holding the defence system where it is now by maintaining or changing the standard of protection, such as seawalls, or emplacing rock armour.

- **Advance the line (AtL)** this involves building new defences seaward of the existing defence line and to reclaim land from the sea.

- **Managed realignment (MR)** this involves allowing the shoreline to move seaward or landward, with associated management to control or limit the effect on land use and environment.

- **No active intervention (NAI)** this involves no further investment in coastal defences or operations.

The SMP describes in greater technical detail as to why these decisions are taken at any one given section of the coast. This process is then repeated for each of the subsequent epochs, the set of images below demonstrate this subtle change in policy decisions over the epochs.

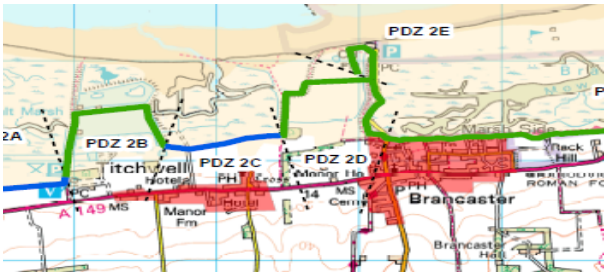

Figure 2: Brancaster area epoch 1 [97].

In Epoch 1. The area around Brancaster has been designated as either Hold The Line (green) or No Active Intervention (blue).

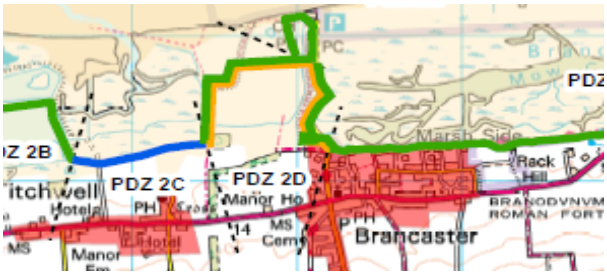

Figure 3: Brancaster area epoch 2 [97].

In Epoch 2 the areas of the designation are the same but the land directly north or Brancaster has also been designated with an additional yellow line, which states that this area would be subject to a managed realignment if the conditions had deemed it so.

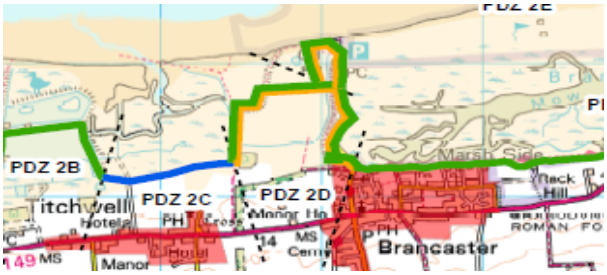

Figure 4: Brancaster area epoch 3 [97].

In Epoch 3 there are no significant changes beyond Epoch 2. This of course could change in future SMPs, if local conditions change. In this specific case of Brancaster the conditional Managed Realignment it suggests is labelled MR3. Managed Realignment is classified in one of 3 types as stated in the SMP.

- **MR1** Maintain the flood defence function of a natural defence with minimum intervention, allowing maximum natural development
- **MR2** Breach the front line defence after building a new defence further inland.
- **MR3** Breach the front line defence, no new inland defence.

As a high level document outlining shoreline defence strategy the SMP seems to be surprisingly robust. Though it is limited by the depth of research into the processes and sedimentation models of any given stretch of coastline [97, 96].

2.2.3 Managed Realignment

Managed realignment is a seemingly radical policy in the arsenal of coastal groups. In these cases deliberate steps are taken to allow the sea to reclaim part of the shoreline, such as the act of breaching a sea wall, or completely re-engineering a portion of the landscape. In these cases it is often the plan to create a new intertidal zone, such as a salt marsh or lagoon that acts as a buffer against further ingress of the sea [101].

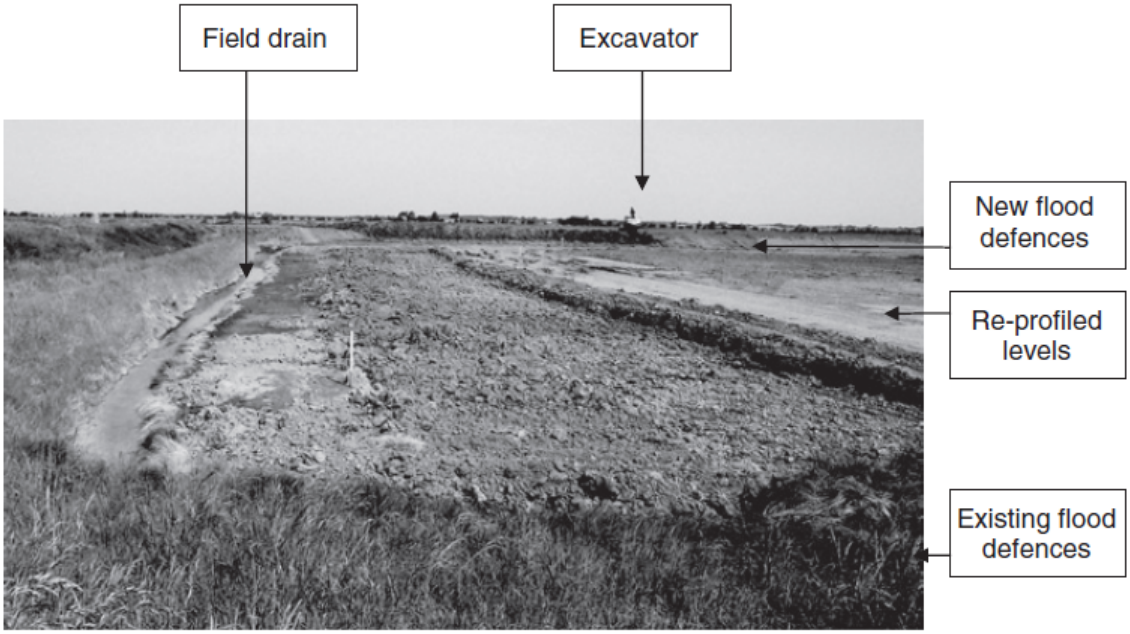

Figure 5: Banked Realignment work progression [71].

Managed realignment is a relatively new approach. The traditional use of hard defences such as sea walls to hold the line has been the mainstay of coastal defence since the Roman era. However the consequences of climate change is making this traditional strategy less and less sustainable [2]. In the UK there are very few sites older than 25 years, so the long term impacts of this approach are yet to be seen, though observations thus far seem to be encouraging. Akinson et el [7] have shown that mudflats created by managed realignment show a striking similarity in biodiversity to their natural analogues [71]. Artificial salt marshes however seem to be a more challenging undertaking, where even decades after the initial formation of the new habitat the plant life is still different to the surrounding salt marshes [7].

2.2.4 Realignment Case Study Tichwell Marsh Bird Reserve

Figure 6: Tichewell Marsh bird reserve location [35].

Tichwell Marsh is the RSPBs flagship reserve is situated in North Norfolk just a few miles east of Brancaster and also lies within the boundaries of super frontage SMP 5, the location of this thesis study. It was built during the mid to late 70s as a wildlife reserve, using a series of sea walls to create a number of enclosed environments of salt water, brackish and freshwater habitats; since its construction it has attracted a sizeable population of birdlife throughout the year.

It should be noted that because of Tichwells location it is part of two SACs, a SSSI, an AONB and a SPA. Incidentally the entirety of the Norfolk coastline is considered to be a Ramsar site (a wetland of national importance). This places Tichwell in one of the most protected areas in Britain in terms conservation. It is for all of these reasons as well the importance of local birdlife that any coastal management project is considered within incredibly strict parameters.

The issue of managed realignment appeared due to the perceived inadequacies of existing coastal defences round the marsh. Indeed after a scoping study was undertaken by Royal Hosking it was deemed that the remaining life span of some of its seawalls would perhaps not even last a year without intervention. The reason for the failing sea defences is blamed on global sea level rise, which leads to a gradual encroachment of the sea. It was noted

that that even with a new sea wall the maintenance of the brackish water habitat in the reserve would not be tenable within 50 years.

Figure 7: Tichewell Bird reserve before the managed realigment project took place [81].

The above image is an aerial shot of Tichwell before the managed realignment had taken place. According to the Hosking non technical summary the following actions were undertaken to each of the sea walls. This is stated as verbatim from the document.

East Wall: Breaching of East Wall (to the north of the Parrinder Line) to allow saline intrusion into the Brackish Marsh. Bring the remaining East

Wall defence (in the south east corner) to 1:30 standard where required. No works to the East Wall north of the breach. North Wall: Leave in place with no works and monitor the North Wall and beach and coastline change. Possible removal of North Wall in the future if monitoring shows it is interfering with beach processes.

West Wall: Hold the existing defence line by strengthening and raising sections of the wall (south of the Parrinder Line) to the appropriate standard of defence. Maintain the northern part of the West Wall for public access to the beach and for possible future works to the North Wall.

Parrinder Line: Strengthening and raising of the Parrinder Line to provide the appropriate standard of defence and act as a primary defence. This would include redesign of the Parrinder hide and maintaining the double-wall access to the hide (to avoid bird disturbance). Any works to the wall to be preferentially to the north of the line to minimise any damage to the remaining freshwater part of the reserve.

Freshwater boundary: No proposed works. Maintenance only.

South East Corner: Minor works. Maintenance and seawall linking into higher ground.

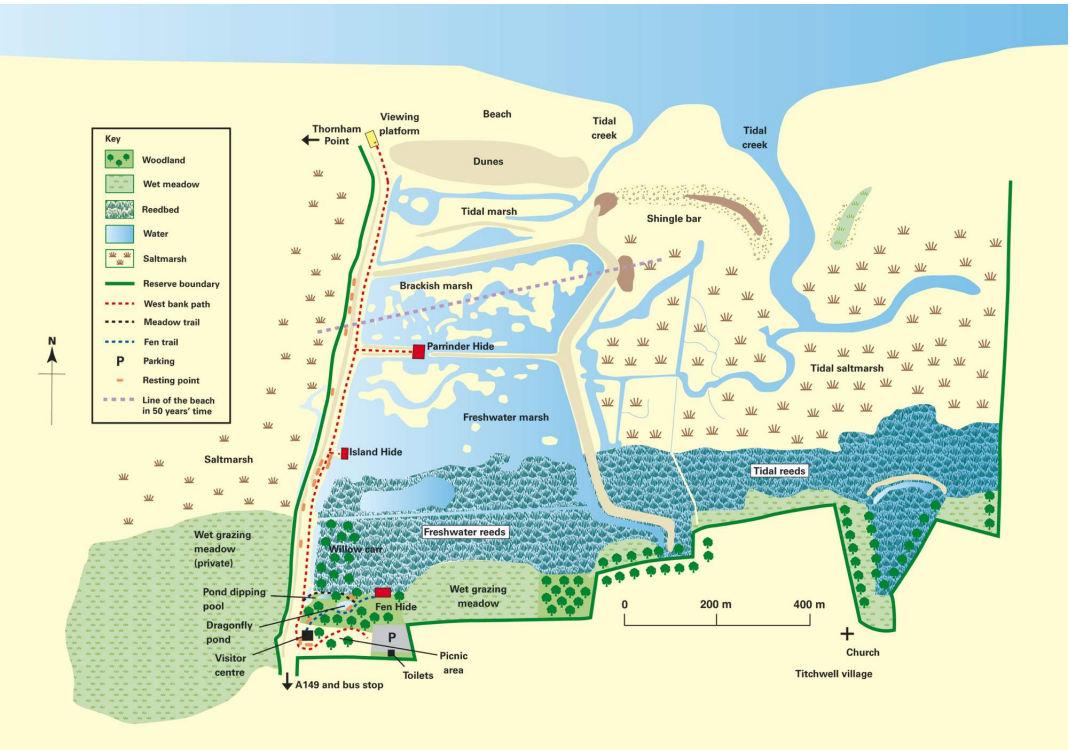

Figure 8: Tichwell bird reserve after the management realignment scheme [8].

Figure 8 shows the marsh after the completed work. It does not show it, but a sluice gate has been added to the east wall by the brackish marsh.

The main thrust of the work was to prepare for an advancing sea front which will be assumed to have advanced to the brackish marsh within 50 years time. Because of this the Parriender wall which separates the fresh water habitats from the brackish and salt water habitats has been considerable strengthened. In addition the RSPB has taken advantage of this re profiling to add a new hide to the wall.

Due to the protected nature of the site environmental impact assessment was undertaken that outlined a number of strict parameters that the re-profiling work had to be undertaken within.

Work on the realignment was completed in 2010 and seems to have been a tremendous success, providing strong mitigation for sea level rise in the future. However the work is based on the assumption that the growth of

the Scotlhead Island spit would continue westwards, providing some form of protection to Tichwell. This is based off a series of sedimentation models undertaken by Hosking, however sedimentation modelling is extremely difficult to do accurately due to the number of variables required, and unpublished work by the university of Kingston shows that westward march of Scolthead's spit is by no means certain [81, 80].

2.2.5 Holding the Line

Holding the line has been the default position for coastal defence in the UK since the Roman era [2]. It has developed a number of strategies to attain this, many of which involving "hard defences", as of such the hold the line policy has become increasingly expensive to maintain due to the increasing pressures of climate change. With a focus on "hard defences" hold the line is an engineering centric approach.

Figure 9: **Groynes:** Walls running perpendicular to the coast to retard longshore drift. They are cheap to produce at around £15 000 each but excessive use can lead to Terminal Groyne syndrome where groynes radically reduce sediment transported by long shore drift down shore. The lack of available sediment down shore can dramatically increase erosion rates [42].

Figure 10: **Sea walls:** Usually made of concrete, seawalls are the most robust of hard defence strategies, resisting erosion and deflecting hydraulic pressure, but they are also the most expensive at approximately £10 000 a meter. Due to their cost they are employed only where necessary (e.g. urban areas)[42, 20].

Figure 11: **Reverments, Rock armour and Gabions:** Collectively termed as armour, these options are cheaper if less sturdy alternative to seawalls. Revetments re profile the sea front dissipating wave energy. Whilst Gabions and Rock armour provide a physical barrier and baffle against wave action. All these armouring options are moderately expensive at around £4 000 per meter [42, 70].

Figure 12: Typical example of rock armour found in the UK.

Figure 13: Typical example of gabions found in the UK [42].

Figure 14: **Offshore breakwater:** Large structures often made of concrete or boulders built some way from the shore. They are expensive to make at around £4,000 - £10,000 a meter but require very little maintenance . They diminish wave power, and thus reduce erosion but can conversely speed up sedimentation leading to the creation of sandbars, and occasionally tombolos (gravel or sandy beaches that link to offshore structures) They can also form rip tides (strong currents leading from the shore seawards) which can be extremely hazardous to beach users [42, 84].

Soft approaches: Softer approaches either replace lost beach material with new sediment, or encouraging vegetation on existing sand dunes to stabilize their positions. Beach nourishment however can be relatively expensive costing up to £200 000 for every hundred meters and often can have a rather short life span. Issues with cost and lifespan can be exacerbated by the material chosen, if the grains are different to the natural sediment they will behave differently in the face of hydraulic and wind energy. Beach nourishment is a way of reliving the symptoms of erosion but does not deal with the cause, but if used in conjunction with other management strategies it can be highly successful [42].

2.2.6 Advancing the Line

Advancing the line is the policy of land reclamation seaward of the current coastline. It is a relatively rare stance to be taken by coastal groups, one can only assume because of the costs involved in any land reclamation scheme. Land reclamation can be achieved by a number of means depending on the outcome desired.

Beach nourishment: where imported beach material is used to replace lost sediment. This is also sometimes used in Holding the Line strategies but can be extremely expensive at £20,000 per meter. In addition if the incorrect material is used it can be washed away relatively quickly. [42].
Seawall cut-off: By expanding seawalls round a shallow marine area it can then be drained and furthermore built upon. The Netherlands is famous for its series of dykes, and extensive land reclamation, in fact approximately one sixth of its totally landmass is reclaimed land.
Artificial Island creation: Artificial islands are generally no small undertaking, they require large scale engineering works and the investments required are usually vast. Generally the land is created by large scale feeding of sediment into the area, which is then protected through use of any number of hard defences. An example of this would be the construction of The Jumeirah Palm, which is 5 kilometres in diameter and made from some 70 million cubic meteres of sand.

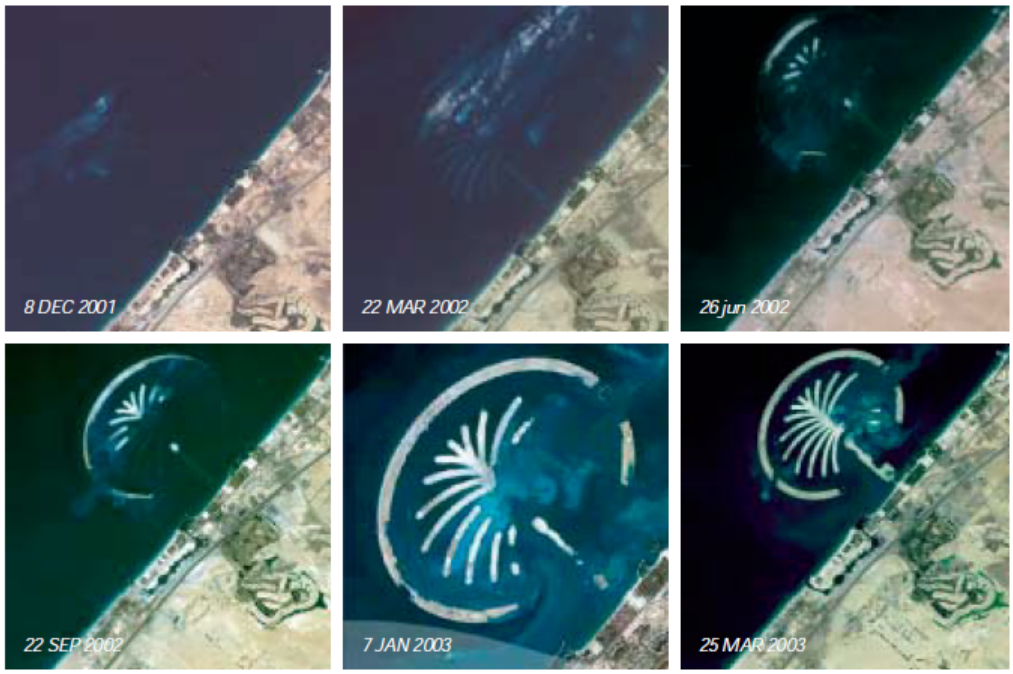

Figure 15: Time lapse photos showing the growth of the artificial island via sediment upwarping [50].

The relatively small tidal range of this near equatorial region matched

with the formidable offshore breakwaters built around it means the island is well protected from erosion. Figure 15 shows how it was constructed via sediment feeding over time.

2.2.7 Concluding remarks

The grand achievement of the Palm Island and even the reworking of RSPB Tichwell clearly shows how plastic the relationship humans have with the coast and the capabilities of coastal management when the resources are made available.

However resources are limited and therefore prudent decisions are required. In the section 2.4 crowd sourcing is discussed as a way to resolve complex problems, considering the admission in parts of that North Norfolk SMP that areas require further study before and informed decision can be made this may be a potential solution or at least a source of information and ideas.

The approach of policy making by coastal groups is based heavily on rules and goals which lend itself well to the construction of game like environments.

In the light of emerging environmental issues such as sea level change due to climate change, increasingly innovative approaches to resolving environmental issues must be considered. The radical steps of managed realignment that abandons the idea that holding the line is necessarily the best option shows the breadth of decisions policy makers are willing to give the green light.

Perhaps it is time for them to reconsider the way in which the consultations are made?

2.3 The Digital Earth

Geographic thought goes through three distinct phases as the relationship between humans and their environment matures. Exploration, Description and Explanation. First humans explore their environment, to become familiar with it, then they describe it, map it collect samples of things within its bounds. Finally humans seek to explain the collected data, they create hypotheses and theories concerning the new area and finally seek to uncover real truths within the information [60].

The Digital Earth is the ultimate expression of geographic thought. It has emerged through our desire to explore our world and space, to map it,

collate its data and extrapolate it all into a single unified construct.

2.3.1 The Digital Earth

The genesis of the Digital Earth came from a talk at the California Science Centre in Los Angeles on the 31st of January 1998. In this talk the former Vice President of the United States Al Gore proposed the creation of virtual reconstruction of Earth, the Digital Earth.

Not simply would this be a to scale reconstruction of Earth, but it would be suffuse with spatial data. At any point the user could interrogate world, in the talk Al Gore suggested local crime statistics, biodiversity and historical records as examples. In summary the Digital Earth was described as "A multi-resolution, three-dimensional representation of the planet, into which we can embed vast quantities of geo-referenced data". Assumed applications included:

- International diplomacy and border disputes.
- Fighting crime.
- Preserving biodiversity.
- Predicting climate change.
- Increasing agricultural activity.

A modest proposal was put forward (in relation to the total vision of the Digital Earth), that the "silos of Landsat images" be put to good use to create the first generation of Digital Earths, a virtual reconstruction of the planet be made at the resolution of one meter square. [38].

This ambition was met in 2005 with the release of the now ubiquitous Google Earth.

2.3.2 Prehistory

Earth imaging technologies provide the back bone of the Digital Earth Technologies and it officially began in 1972 with the launch of the Landsat Satellite, however the first images of Earth were taken during the era of the Apollo space program with the most famous of these images Earthrise was taken by Bill Anders from the orbit of the moon during the Apollo 17 mission. It is

consistently chosen by Life magazine as one of the 100 most important photos taken in history, and described by the acclaimed photographer Galen Rowell as "the most influential environmental photograph ever taken" [63, 57].

Figure 16: Earthrise, the first ever photo taken of Planet Earth from the perspective of another celestial body [63].

2.3.3 Landsat

The Landsat program is the longest continuously running earth imaging survey program providing the bulk of the images for Geo-Browsers and other first generation Digital Earths so far. Landsat 1 was launch in 1972 only a few months before the famous Blue Marble picture was taken by the crew of Apollo 17, and although the program was not considered to go live until the launch of Landsat 2 in 1975 the quality of the images produced ensured its popularity and success within the US congress.

The current iteration of Landsat used by NASA and its affiliates is Landsat 7, which was launched in 1999, its original specifications was to have a 5 year shelf life but the engineering process created a far superior product. Landsat 7 is a joint NASA and USGS venture.

The Landsat Data users handbook describes the mission features as following.

- Data Continuity: Landsat 7 is the latest in a continuous line of remote sensing satellites which have been in operation for 38 years in total.
- Global Survey Mission: Taking up to 250 images a day to build and refresh a global archive of land images every 16 days. With the aim for available images to be sun-lit and cloud free.
- Affordable data prices: 0R pictures are free to download by all, made available within 24 hours of being taken. Greater resolution 1R pictures can be commissioned by private or public bodies at cost.
- Enhanced Calibration: Allowing the ETM+ (Enhanced thermal mapper plus) technology an unprecedented level of accuracy.
- Responsive Delivery: To provide products in 48 hours.

Landsat 7 has a 378 gigabit solid state recorder for created images, further added to this is an X-band transmitter. X-band transmitters are often used within LEO satellites as a broadband data beam transmitter. In the case of landsat 7 it has a 75Mbps bandwidth.

To create a consistent set of images Landsat seeks to take a photos of each picture centre at the same coordinate each 16 day refresh. To do this it orbits in a Polar orbit (where it travels pole to pole as opposed to parallel to the equator) in what is known as a swath pattern. As shown in figure 17.

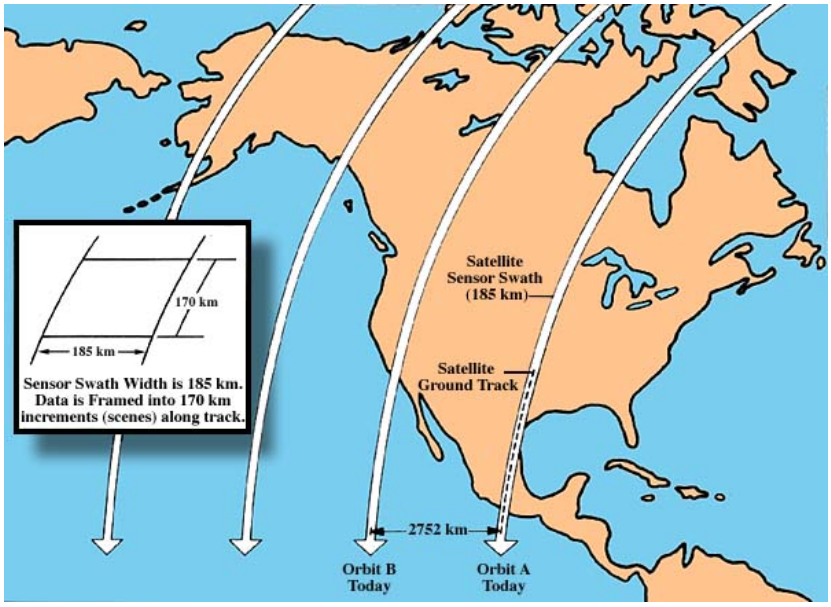

Figure 17: The swath pattern of orbit for Landsat, designed for maximum coverage of the Earth's surface.

This creates a tessellated movement pattern if mapped onto a Digital Earth or GIS software, and shows that there is little drift from each of pass of Landsat, ensuring a drift of no more than 250 meters of each image's epicentre. Shape files, and path files are available through the landsat website [64] allowing for potential users to time their image requests with Landsats path.

2.3.4 The Growth of Digital Earth Technologies

The Digital Earth Vision today has evolved from Al Gores original description. Since the time of its inception The Digital Earth has had a number of dedicated participants. The most notable of whom is the International society for The Digital Earth whom with financial backing from the Chinese government passed a declaration for the active pursuit of the creation of the Digital Earth in earnest [14]. In addition much of the foundational work and conceptualizing of how the Digital Earth should be created was envisioned by contractor SRI international as part of a DARPA contract. SRI stressed that for the Digital Earth to be truly successful it must embrace a philosophy of openness and open data. Communication between developers

will have to be transparent and candid for efficient transferral of ideas in effectively what would be a whole new suite of technologies. Furthermore the success of the Digital Earth project relies upon access to vast swaths of spatial data; therefore creators of this data must be encouraged to make their data open and accessible. In many cases the creator of this data is the public sector and fundamental policy shifts about making data open will be necessary [88]. From these conceptualisations the following technologies have emerged or been developed.

In 2008 an editorial piece in the International Journal of Spatial Data Infrastructures Research provided a retrospective of the Digital Earth vision, detailing its successes and failures. It concluded that the original vision of a single all encompassing Digital Earth was perhaps misplaced, emerging Digital Earth technologies are commonly applied in a problem orientated fashion, and this focus on specific issues has created a demand for specialised Digital Earth platforms [17]. An example of this trend to diverging requirements comes from a study by Brown et al in 2004 [12], where realistic simulations of the effects of coastal erosion were modelled using flight simulation software to add an extra dimension of realism. This more realistic visualisation found great enthusiasm with the public, but professionals such as coastal engineers found it generally unhelpful since they demand a visualisation with more abstract representations allowing for more detailed analysis of the region.

Digital Earth technologies have seen great progress, the most obvious of this being the development of Geo-Browsers. Geo-Browsers are the first generation of Digital Earths, a virtual globe created through satellite imagery and populated with an array of spatial data; Google Earth [35] is the most well known of Geo-Browsers but the list also includes NASAs World Wind [65], Microsofts Bing maps [61] and Open Street Map [69].

Whilst Geo-Browsers have primarily been created by private sector organisations SDIs (Spatial Data Infrastructures) have been developed almost exclusively by the public sector, and whilst SDIs have seen significant development it has been noted that inefficiencies in the public sector has lead to some delays in SDI creation [17]. Where Geo-Browsers seek to visualise spatial data SDIs organise and reference it the SDI Cookbook refers to a SDI as the following: "The term Spatial Data Infrastructure (SDI) is often used to denote the relevant base collection of technologies, policies and institutional arrangements that facilitate the availability of and access to spatial data. The SDI provides a basis for spatial data discovery, evaluation, and application for users and providers within all levels of government, the commercial

sector, the non-profit sector, academia and by citizens in general." [33]

A prime example of a current SDI is the European Union's INSPIRE project which will allow the public sector organisations across Europe to share and access spatial data, as well as providing the general public more ready access to this data. INSPIRE also includes a number of EU legislations which makes it easier to share, and access data throughout Europe [24].

The addition of KML, an XML derived markup language as both de-facto and OGC standard language for all Geo-Browsers in 2007 has brought much needed standardisation to the auspice of Digital Earth technologies. The benefit of this standardisation means that the sharing and referencing of spatial data is significantly easier [106].

Considering the refresh rate of earth images, geo-browsers and other digital earth technologies are an excellent medium to study coastal erosion, Digital earth technologies can record and monitor these changes.

2.3.5 The future of Digital Earth

Although wildly speculative, the potential future iterations of the Digital Earth can be found in Science Fiction, the science fiction classic Snowcrash itself has been attributed for the inspiration of Google Earth [73]. Perhaps the most interesting concept of where Digital Earth technologies can lead is a recurring theme in many novels by Dan Simmons where augmented reality is combined with Digital Earth technologies. The author describes this as the All Net, an augmented reality technology which overlays over the users vision. It effectively allows the user to interrogate the environment in a similar manner to Al Gores use case of the school girl in the original Digital Earth vision, but to a degree of much greater complexity, going down to the sub atomic scale if the user wishes it [83].

What ever the end point of the Digital Earth technology may be, the current trends seem to be away from a single Digital Earth, to large SDI initiatives run by the public sector and increasing standardisation. The consequence of this is that created spatial data can be displayed, viewed and manipulated with increasing ease by the general public. The London Data Store which freely publishes public sector data under the open government licence has seen an explosion of people who use its data to create mash up maps and mobile applications, all taking advantage of the spatial nature of the data [39, 99].

Furthermore as much of this project will seek to explain, collaborative

efforts of online communities supplied with the right tools can be prolific with their output. Digital Earth technologies are an optimised tool to present vast quantities of data in an easily accessible way, and thus are a great opportunity for any community that wishes to explore, present, or analyse spatial data or regional issues.

2.4 Collaboration and Crowd Sourcing

In the aftermath of the climategate scandal where the CRU of EAU was accused of falsifying data to support the claim of anthropogenic climate change the independent review panel exonerated them of all accusations. However the panel recognised that the CRU lacked transparency and this fuelled scepticism, they recommended that the pubic should have full and open access to the data at any time [11].

By providing a forum to discuss specific data sets, and having the correct tools to interrogate the data correctly a community can emerge. In this section the concept of gameification will be explored as a novel way to engage a community around, as well as finding examples of successful communities that exist on internet that maintain exceptional outputs of content on a voluntary basis.

2.4.1 Collaboration within online communities

The internet by its very nature allows anyone with access regardless of their location to indulged their passions with like minded individuals. Often it is the case within these communities that they create unique content associated with their chosen topic.

The very first online communities emerged in ARPANET the predecessor of the internet. These were bulletin boards that allowed an individual to create an initial post and for a thread of responses to stack up below it, very much forming the blueprints for how modern internet forums. However it was the the development of the world wide web (www) that and the subsequent emergence of WEB 2.0 that has seen the rise and rise of online communities.

An online community is a rather nebulous definition. They are as diverse as human taste and technology allows, but to provide a framework in which online communities can be considered we can break them down into the following [72].

Gaming communities: These communities collaborate together to further their enjoyment and competency within a single or multiple games, clans and guilds found in MMOs are an excellent example of this.
"Glocal" communities: in 2002 the sociologist Barry Wellman described the phenomena of glocalism, where local communities such as churches, clubs and schools create an online presence to compliment the real world community [9].
Hub based communities: These communities tend to focus on a specific site which acts as a hub for their shared interests. These types of communities tend to reflect membership by interest as opposed to geographical location.
Dispersed communities: These differ from hub based communities due to their lack of centralisation. The community is linked more by philosophy or interest rather than a hub and thus will use multiple resources to communicate with one another. The hacktivist group known as Anonymous would be the perfect example of this type of community.

2.4.2 Collaboration as content creation

Online content creation is exemplified by the open source software movement, which is defined as software where the code is made freely available to the public who can modify it, adapt it , although this software is free, there is strong ethic in these communities that the source of any used code is always acknowledge.

Linux is probably the best known example of open source software [56]. The initial developer Linus Torvalds found that during an unrelated project he had inadvertently created a near complete OS, he posted his work on a bulletin board not wanting his efforts to go to waste and his project seized upon by the community [100].

In 1998 a number of confidential Microsoft internal documents were leaked to the press, these were designed to impress upon the reader the threat open source software posed to Microsoft. It appears that the concerns of these documents were well founded, open source software projects often rival proprietary based ones, Android software for mobiles was created under an open source philosophy [94], and the most common server side software to date is augmented Linux [104].

Game modding has in recent years grown to become a hot best of collaborative content creation. These communities create add-ons for existing games, or change the way in which they play, some are tremendously popu-

lar. DayZ a survival horror mod set in the aftermath of a zombie apocalypse completely reinvigorated the sales of ARMA2 the military shooter game it was based on. ARMA2 by Bohemia Interactive was released in 2009 and saw only modest sales [46, 40], but with the realise of DayZ in 2011 the popularity of the mod such that daily active users of ARMA2 surged by 847% and over 300 000 new copies had be sold within two months [74].

Increasingly games developers provide the tools for the community create content for the game. Valve's Steam workshop allows players to created customisable hats for Team Fortress. As whimsical as this may seem, the hat economy that emerged from the workshop market has been hugely successful[62], Gabe Newell the founder of Valve stated that a single individual has made in excess of $500 000 though this hat economy alone [67].

2.4.3 Collaboration as problem solving

Given the appropriate tools and communication resources it appears that online communities are adept at self organising, this self organisation lends itself well to problem solving.

MMOs appear to be an endless fountain of people collaborating to overcome incredibly difficult problems. Often these games have mechanics built in so individuals can organise into guilds or clans, this allows them to pool resources and knowledge to overcome the challenges of the game world.

A common example of this are "raid bosses", powerful enemies that require the collaboration of up to 40 people in some cases. They are known for their difficulty and often require careful coordination within the guild to be defeated. In the MMO Final Fantasy 11 by SqureEnix [21] the raid boss Pandemonium Warden was deliberately designed to be unbeatable. This did not deter guilds from trying with one guild fighting the raid boss for 18 hours non stop, their guild member Sylphet recounts the reason they finally gave up the attempt.

> ‘‘People were passing out and getting physically ill. We decided to end it before we risked turning into a horrible new story about how video games ruin people’s lives.’’ Sylphet 2008 [18].

However within a month through a quirks of a specific item within the game mechanics a rival guild called Absolute Virtue defeated this boss in less that a minute [6].

The items dropped by these enemies are often much more desirable because of their rarity, and this presents a difficult economic problem to the guild. Namely who should receive the item?

The proposed solution was Dragon Kill Points (DKP) (named after the dragon raid bosses in Everquest where this system was developed). Since raid bosses respawn overtime, this allows guilds to mount successive raids and with each raid boss killed participating players would each receive 1 DKP. DKP could then be used as a currency within the guild to bid on the items dropped by these bosses. The solution is simple and elegant, it encourages active support of the guild to reward the individual and as of such it is considered the default way to deal with the problem of loot within nearly all MMOs to date [26].

Crowd sourcing also has seen transformation success in the real world. The "Goldcorp Challenge" being perhaps the most famous example of this, in 1999 the new CEO of Goldcorp Rob McEwan found the company beset with rising costs and dwindling reserves [44]. Incidentally he attended a seminar at MIT discussing the open source development of Linux, inspired by the success of this open technique he launched the "Goldcorp Challenge". The company released the entirety of its exploration data and offered a pot of $575 000 to be shared between the individuals who could correctly identify unexploited gold veins [107].

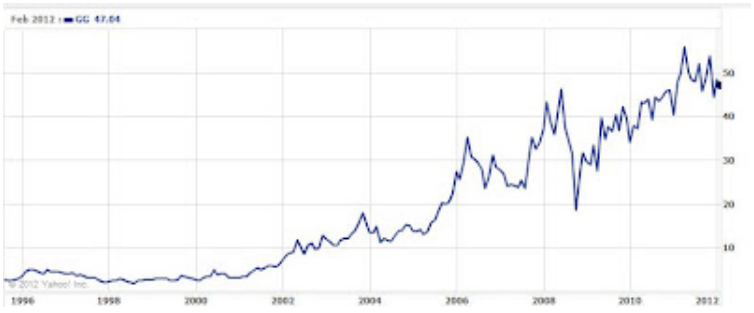

Figure 18: The rise and rise of Goldcorp's stock [59].

The results of the crowd sourcing contest were staggering. 110 new sites were discovered providing to the company an additional 8 million ounces of gold and turning it into the second largest mining company in the world. The chart below shows the incredible change in company Stock value after the issuing of the Goldcorp challenge [13].

The Goldcorp challenge is on of the first examples of successful crowd sourcing through the internet, and since its inception many variations on this has been implemented including Amazon's Mechanical Turk where a company can outsource simple jobs to the crowd sourcing site [3].

2.4.4 Collaboration as science

Crowd sourcing appears to have success in scientific research too. Galaxy Zoo [108], a website created in 2007 as a method to crowd source the job of classifying galaxies since the work load far exceeded even the most dedicated research team.

Galaxy zoo was created in 2007 and as a way to crowd source the massive task or organising and classifying galaxies observed by the Hubble telescope. The repository of images of individual galaxies that required classification ran into the millions, a work load far exceeding even the most dedicated research team. The solution was to crowd source this process, through the creation of the Galaxy Zoo website where web users can classify a series of observed galaxies.

Within the first day some 70,000 classifications were made, which expanded to over 50 million different classifications by the end of the first year alone. A strong community of almost 150 000 participants has emerged around galaxy zoo, ranging from the interested lay person to other professional astronomers. This community has proven itself a valuable research tool identifying a number of previously unseen celestial phenomena including Green Pea galaxies and Red Spiral Galaxies.

Figure 19: Examples of green pea and red hot galaxies as the users would have seen them during user classification [52].

Foldit is another highly successful attempt to use crowd sourcing to further scientific research. Foldit is a game that was released by a collaboration between the biology and game science departments of Washington University. It is a game that revolves around the process of folding proteins, whilst seemingly arcane the way a protein folds and compacts affects the way it interacts chemically, making research into this topic highly desirable.

Until recently the main methods of determine optimum ways of folding proteins was performed by computer simulation or direct observation, however it was found that both of these processes were expense and time consuming.

Foldit was conceived as a way to take advantage of humans' natural pattern finding abilities and to crowd source the problem at the same time, since previous methods to uncover optimum ways to fold proteins were time consuming and expensive.

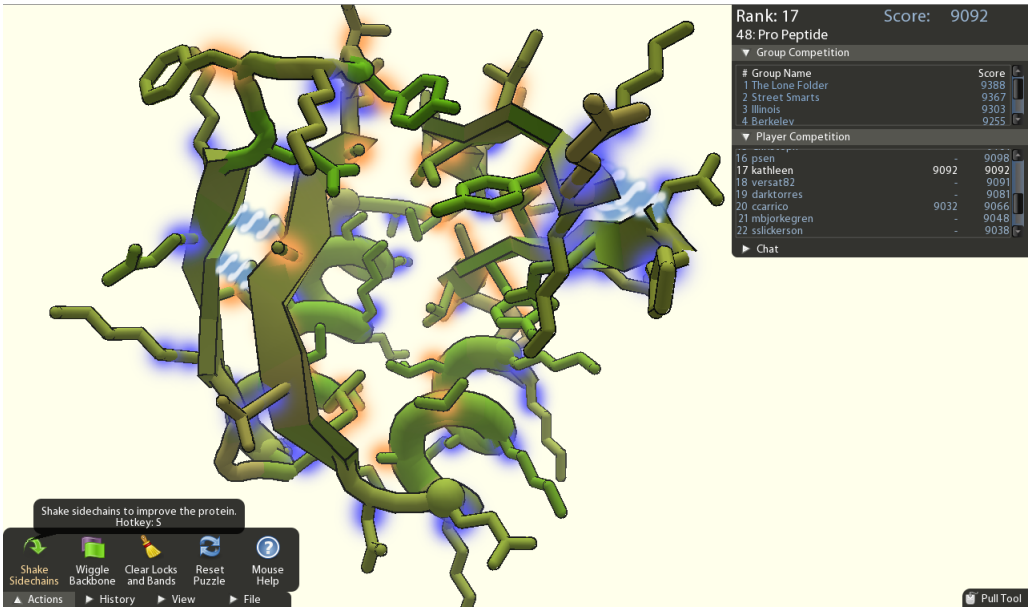

Figure 20: FOLDIT, protein folding has never been so much fun [103].

The game user interface is shown in figure 20, the game play involves manipulating the protein to find the optimum way of folding that protein. The player's efforts are then recorded and scored on a leader board to foster a sense of competitiveness within the community.

The game has proved compelling and like Galaxy Zoo it has proven hugely valuable in the terms of scientific research. Most notably was the work to uncover the structure of Mason-Pfizer monkey virus, which is considered a link to the AIDS virus. Since its discovery the problem had eluded scientists in the field for almost 15 years but upon release into the game it was solved in less than 10 days [103].

2.5 Gamification

While discussing collaboration, it is rather noticeable that many of the examples are based around online games, MMOs in particular seem to be fertile ground for forging communities that pursue extremely complex problems. it appears that there is something inherent in game mechanics that are engaging to both groups of people and individuals.

2.5.1 Gamification as a method of engagement

In the 2011 book Reality is Broken the Author Jane McGonigal posits the question as to why so many people find virtual worlds so much more engaging that reality? [58] The answer she suggest is that certain mechanics exist within games that people find irresistibly engaging, furthermore she suggests that these mechanics are not exclusive to games alone. In fact game mechanics have been used in the real world for some time, often in the form of incentives such as store loyalty cards to change an individuals behaviour.

As to why people find games so engaging Jane McGonigal quotes the philosopher Bernard Suits who invokes the core nature of any game.

`‘‘playing a game is a voluntary attempt to overcome unnecessary obstacles’’.` Bernard Suits 2011.

She suggests that it is these unnecessary obstacles that humans find so appealing, they provide a set of challenges to overcome, which provides a sense of achievement when the goal is met. This is as true for games such as football, golf and cricket as it is to computer games such as Civilzation [32], World of Warcraft [22] and Angry Birds [23] [58].

2.5.2 Mechanics of Gamification

Whilst the unnecessary obstacles provide the core of what a game is, games have in numerable mechanics that further incentivise the player into engaging further with the game. Many of these mechanics are used outside of realms of games to provide boosts to commercial ventures or to attempt to engage a community for specific purpose.

Skinner box incentives: Pavlov showed with his famous dog experiment, how behaviour could be conditioned through stimuli. Burrhus Skinner took this one step further to create the operant conditioning chamber (the Skinner box). This chamber is a small box in a single bird is housed, also in the box is a single button and a seed dispenser. Each time the button is pressed via the bird pecking it seeds are dispensed. The bird quickly recognises the relationship between the button and the dispenser, and thus its behaviour changes to press the button repeatedly to receive the reward of food. The impact of these so called Skinner boxes was to show that unlike Pavlovs dogs in which outward stimuli incentives behaviour, the Skinner box relies on active participation of the subject to reinforce their own behaviour to achieve a desired result.

Figure 21: The Skinner box

Further studies using Skinner box techniques have found that eventually physical desires such as want for food and sex are eventually sated. However similar experiments with humans of input and reward have found that our desire for more conceptual rewards (such as money and status) have a much higher threshold of satiation. It is particularly this discovery that computer games and game mechanics latch onto.

Time/location based incentives rewards an individual for performing a desired action at a specific place/time (for example happy hour at a bar). Progress based mechanics such as completing a progress bar tap into the more conceptual rewards that humans find much more irresistible and thus are used to great success within games like World of Warcraft. Whilst status based incentives such as achievements found in Xbox and Playstation provide a why for individuals to compete directly with their peers.

It should be noted that many game critics consider these mechanics as ultimately un-fulfilling. These mechanics are designed to compel a player to return to the game time and time again, but of course as it has been pointed out, compelling is not necessarily fun.

2.5.3 Summary

People will readily collaborate within online environments. The internet allows for people to socialise by interest (and obsession) in leau of geography, it allows for like minded individuals to exchange ideas, and creations within

communities of their own peers with ease. Further more since these communities are inherently niche and celebrate that chosen interest, the peers within it will encourage creation of content and critique it effectively.

To effectively encourage the growth of these communities a hub within which they can gather around must be suitable for purpose. The history of the internet has shown that as soon as the tools are available to create these forums for discussion, communities surrounding that subject rapidly appear. It is very much a "if you build it they will come" situation. However to allow these communities to collaborate effectively, the tools to communicate and collaborate on projects mush readily exist.

3 The North Norfolk region and The Coastal processes effecting it

In the national SMP The North Norfolk coast is designated as SMP5, a section of coastline running from Hunstanton eastwards to Kelling Hard. This section of coastline is dominated by two large sedimentary structures, Scolthead Island, and Bakeney Point spit.

Much of the North Norfolk coastline is a fiercely protected area, with much of it having the status of a SSSI, SPA, AONB or a Ramsar site; this is largely due to its extensive salt marsh habitats, and the large bird life population that depends on this natural environment [97, 96].

In recent years the North Norfolk coast has seen encroachment of the sea, future predictions of sea level change influenced by climate change predicts that by 2055 sea level will have risen some 225 millimetres [81, 80].

3.0.4 Geological Overview

The geology of the North Norfolk coast is relatively simple; it has extensive bedrock of Cretaceous era chalk, overlain by quaternary drift deposits. The Chalk bedrock was deposited during the Cretaceous period, some 62 to 132 MA and will have formed in a shallow marine setting lacking any sediment input from a terrestrial source such as a delta or river mouth. This can be deduced from the homogeneous nature of the chalk, being almost uniformly formed from marine microfossils.

There is a sharp unconformity that denotes the boundary between the Cretaceous chalk and recent Quaternary drift deposits. It has been deduced

that both the erosion event that removed all the younger rock layers above the Cretaceous chalk and drift deposits occurred due to glaciation brought on by successive ice ages during the Pliestocene [31].

The Glacial till making up the drift deposits is extremely diverse in its composition due the large distances it has been transported from, its morphology tends to be fairly consistent with large irregular grain sizes occasionally interspersed with boulders. The Till also tend to be poorly consolidated, leaving it more susceptible to erosive forces. The region's famously flat profile is due to the lack of origenic mountain building events.

In terms of sedimentary structures both Scolthead and Blakely Point are young and dynamic, receiving the bulk of their sediment from the Thames. This area is classified as UK sediment cell 3 and can be seen in figure 22 [97, 96]. Its designation as SMP5 is due to the Kelling edge of SMP5 coincides with the North Norfolk drift divide. A drift divide being a point of the coastline where the directions of long shore drift diverge (as shown in figure 23). the Kelling hard drift divide is east/west, hence the westward trend of North Norfolk's sedimentary structures [97, 96].

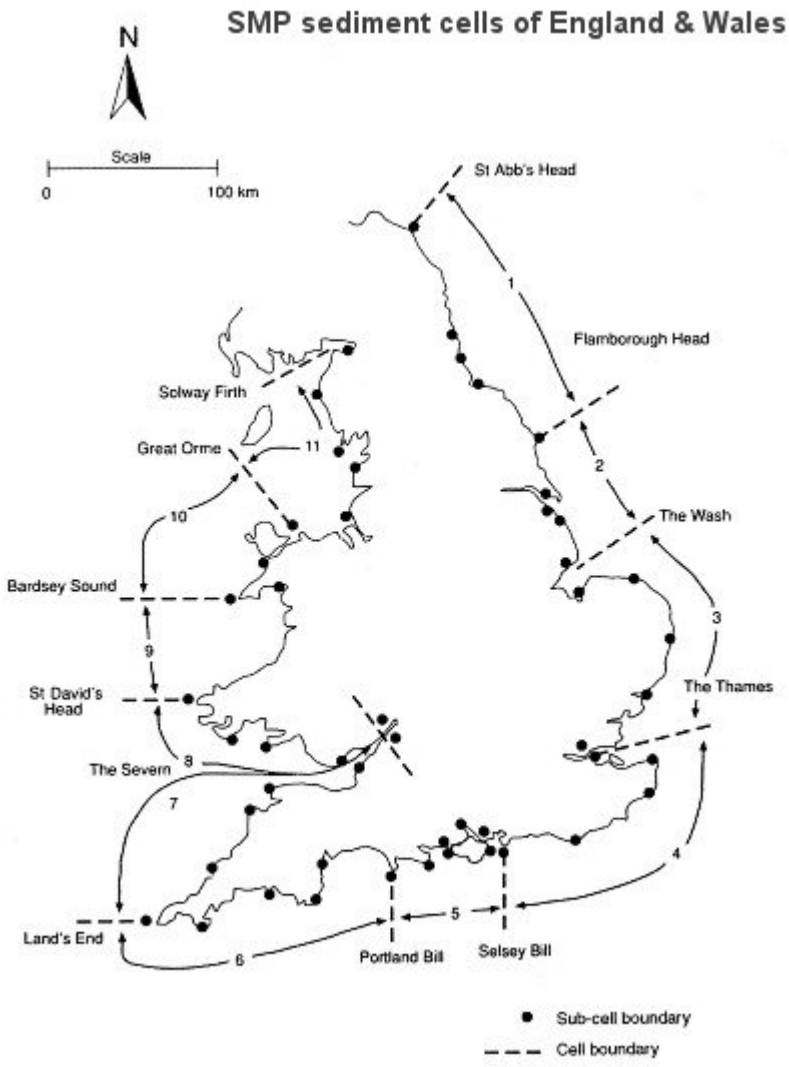

Figure 22: UK sediment cell systems [68].

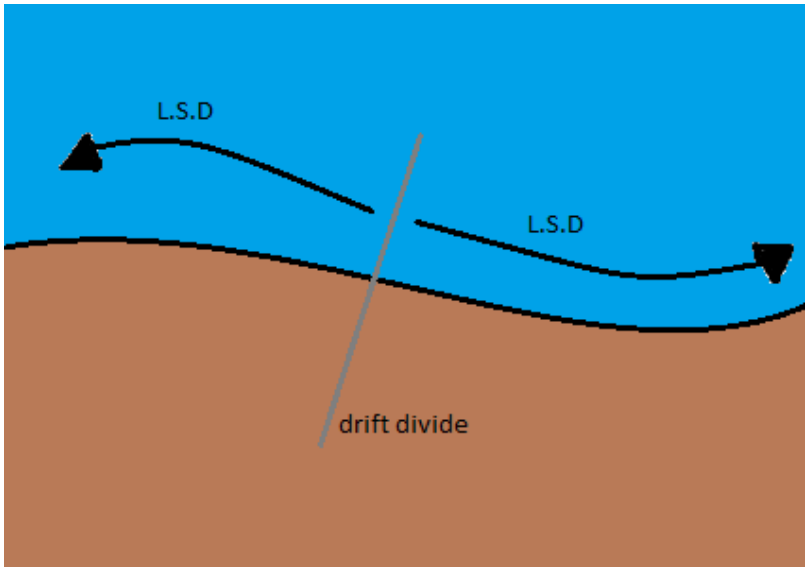

Figure 23: A brief pictorial description of the nature of a drift divide, where the direction of long shore drift diverges.

3.0.5 The Scolthead Island Local

Figure 24: The Scolthead Island local [35].

The coastline running from Tichwell Marsh in the west to eastern most edge of Scolthead island is dominated by the Scolthead barrier island and spit; salt marshes make up much of the habitats around the island and to its south. There is also an intertidal lagoon in its landward shadow, once the tide has receded it leaves a series of tidal creeks and mudflats.

The town of Brancaster Straithe and Burnham Overy Staithe have access to the North sea directly via tidal creeks or the lagoon, much of these town's economies are dependent on this direct access to the sea.

The last important area of note is the Royal West Norfolk Golf Club, this is an extremely prestigious golf course that enjoys royal patronage. However it finds itself in an increasingly precarious position as sea ingress has eroded away much of the surrounding dune system to the point that even the club house is defended by little more than ten meters of land and a seawall.

3.0.6 Scotlhead Island features

Scolthead can clearly been seen a series of spits that trend westwards. Within the safe protection of the spits leading arm salt marshes form. Figure 25 is annotated to show this in greater detail.

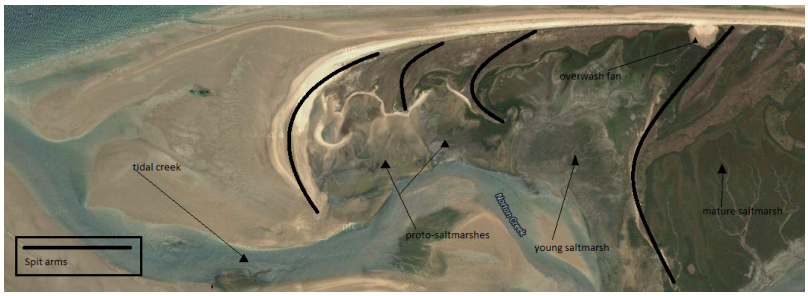

Figure 25: Geomorphological structures found on the leading edge of Scolthead island [35].

Another important feature to be found on the island are a series of over-wash fans. These are formed by storm surges that break through the dunes depositing a large amount of poorly sorted storm sediment in a typical fan shape. Over-wash fans can eventually lead to discordances in the island geomorphology that erosion by wave action exacerbates to break it up or create channels through the island to the lagoon behind it.

3.0.7 Scotlhead Island formation

The Scolthead barrier island appears to be an island formed out of successive spit growth, into which salt marshes have propagated and sand dunes have evolved. However limited research exists on the geological history of

Scolthead, so it is hard to conclude definitively how it formed. To further complicate this there are a number of competing theories as to how barrier islands form and the debate on their merits has caused contention [27, 43].
Offshore Bar Theory: Proposed in 1845 by Elie De Beaumont. Sediment is deposited in an area of low energy near offshore breakers, a sand bar then grows through vertical accretion until it reaches the high tide mark. Once clear of waves and storm wash the sand bar then is then colonised by opportunistic plant life stabilizing the bar into a barrier [27, 41, 78, 5].
Spit Accretion Theory: Proposed in 1885 by Grove Karl Gilbert. This assumes that the source of the sediment comes from prevailing long shore drift that forms spit like features along the coastline. Through wave action or storm over-wash, these features are cut off from the mainland, evidence for this can be clearly seen in many hurricane prone areas of the world [27, 41, 78, 5].
Submergent Theory: Proposed in 1890 by William John McGee. This theory recognises variable sea level and attributes barrier formation to changes in sea levels. Submergent theory categorises barriers in the following manner. Emergent Transgressive: Where off shore bars are submerged during periods of high sea level and grow through vertical accretion in a manner similar to the offshore bar theory. Submerged Transgressive: Refers to dues which have been partially submerged during sea level rise [27, 41, 78, 5].

Scolthead Island appears to be consistent with the spit accretion theory, It lies close to the shore and shows numerous cycles of spit formation on its leading edge. In addition Blakeney point shows a structure similar to Scolthead whilst attached to the shore, suggesting a precedent for this type of feature on the North Norfolk coastline. However due to the dynamic and ephemeral nature of Barrier islands further research in to the island's origin would be needed before this hypothesis could be confirmed. Regardless barrier formation seems to be intimately linked with sea level fluctuation, it is therefore entirely possible that the Scolthead island is a relic object from the recent interglacial periods [31].

3.0.8 Scotlhead Island saltmarshes

Most of the landward side of Scolthead is defined by salt marshes. The image below taken from the older eastern end of the island shows the outline of numerous stages of spit growth into the hollows of which the salt marshes are situated; nearly all the salt marshes are characterised by large tidal creeks.

Figure 26: The features found within Scolthead's salt marshes [35].

However the western edge of the island where spit formation is still active younger immature salt marshes can be found. These proto-marshes are typified with much less abundant plant life and a lack of defined tidal creeks.

Figure 27 was taken from the large established salt marsh in the central part of the island. In the centre of figure 27 a large tidal creek is clearly obvious; it cuts deeply through the thick sediment and is thick with vegetation. Figure 28 was taken from one of the younger proto-marshes, there are many similarities but the tidal creek here is very small since it has to cut through very little sediment and the vegetation is much less advanced.

Figure 27: The environment of the local mature marshes found on Scolthead.

Figure 28: The environment of the young proto-marshes found closer to Scolthead's leading edge.

The sediment thickness is directly linked to the salt marsh age. Salt marshes grow through sediment deposited in-situ via high tides. Once its elevation is high enough to become a muddy intertidal zone salt resistant vegetation can take hold that further increases in-situ deposition of sediment. Studies on Scolthead island itself by J.A.Steer and Stodard et. el [90] have shown this link between sedimentation rates and elevation. Once mature the salt marsh will have elevated above normal sea level only to be flooded during spring over-marsh tides which continue to deposit fine sediment. At Scolthead this is commonly fine tidal silt sediment [89].

The study of sedimentation rates of the main hut marsh by Stoddard et el [90] clearly shows that sedimentation on Scolthead happens at a consistent speed. Indicating that any changes in the environmental factors such as sea level could have dramatic effects on salt marshes. However the same study also is at pains to point out the complexity of factors effecting deposition at Scolthead, and to truly understand the growth and nature of the salt marshes her in detail further research is required.

3.0.9 Scotlhead overwash fans

Over-wash fans on Scolthead are relatively minor features, but if allowed to propagate they can dramatically effect the morphology of the island. Once a over-wash fan has been establish the weakness in the foreshore dune can be exploited by wave action causing the eventual amputation of the island's leading edge. Scolthead has three easily identifiable over-wash fans. Two are found close together on the eastern section of the island and separate larger over-wash fan be found breaking into one of the younger salt marshes on the western leading edge.

Figure 29: Overview of example over-wash fans found on Scolthead Island [35].

The over-wash fans on the eastern end of the island are clearly older objects, having much of their surface area reclaimed by vegetation. The over-wash fan towards the east is still relatively intact as a sedimentary object, retaining its original shape and showing no signs of transgressing vegetation. From this it can be concluded that this in comparison to the other fans is a relatively young object. Direct field observations support this, and figure 31 which was taken in 2011 shows how relatively untouched the fan remains.

Figure 30: The largest and youngest over-wash fan found on Scolthead Island [35].

Figure 31: The local environment of the youngest over-wash fan found on Scolthead Island.

It seems unlikely that the fan will lead to the breakup of Scolthead. The salt marsh it has been deposited into may be immature but is established, and the previous examples of over-wash fans on the island has shown that vegetation is more than capable of reclaiming the area.

3.0.10 Scotlhead remarks

The island remains stable, the over-wash fans may have breached parts of the foreshore but the salt marshes in its lee thrive, so it seems that any major changes in the islands morphology are unlikely. However pressures from sea level rise may alter this and the behaviour of the spit on the leading edge. If this were to happen the consequences for the habitations landward of Scolthead are unknown.

To summarise from remarks found within the SMP itself. The Scolthead area requires further research to adequately make informed decisions on future coastal protection policies. Policy making is at a disadvantage in the Scolthead area because of this dearth of information, yet a thriving community exists. If researchers cannot extract information through traditional means the varied expertise of the local populace whom many have intimate

knowledge of the local environment could be harnessed to gain valuable insights.

4 Game Design

4.0.11 The Agile Approach

This project will use agile development, in particular it will be drawing from SCRUM's project management techniques and Extreme Programming's technical methods. However both SCRUM and Extreme Programming are designed for groups and will need slight adaptation since this project is a individual undertaking.

Agile methodology focuses on a short development periods during which a tangible deliverable is produced. It allows for great flexibility and adaptation to requirements before bugs within the software cascade. Agile methodology requires strong reporting, and consistent testing of the software to ensure smooth development of production cycle to production cycle[30].

4.0.12 Requirements and Methods

	Requirements	Solution
Project work overview	Accessibility, ease of adding or removing information.	RBC(Really Big Chart)/BVC(Big Visible Chart).
Project Todo list management	To record list item requirements, and to be able to mark if completed.	Project backlog spreadsheet.
Work flow management	Record items status (i.e. completed/pending), auditable, documented	SCRUM sprint variant. Burn-down chart. DSDM time box variant. Sprint start/finish documentation.
User feedback	To get feedback from real users, or a system to simulate real users.	No solution yet.

Figure 32:

Big Visible Chart (BVC) The big visible chart is a way to present the entire project in a single glance. In SCRUM and XP from the Trenches Henrik Kniberg states the BVC as the best way to present your project backlog [55]. The meta data for the project backlog can be stored else where in more

detail, whilst the BVC provides an instant and accessible visualisation of the projects progress.

Project Backlog Each and every objective for the project will be stored and recorded within a project backlog, an excel spreadsheet that lists each objective in detail. Each entry in the project backlog will record the item ID, the description of the objective, the importance of the task as well as the manner in which it can be demonstrated after completion [55].

Missions/Timeboxes Missions are an adaptation of timeboxes as described in the DSDM methodology [19]. DSDM describes time boxes as a way to sort project goals into short periods of time. A mission is a preselected package of objectives and organises them within a MoSCoW hierarchy. Must haves are designated as primary objectives, should haves as secondary, could haves as tertiary and would like to haves as quaternary. The mission document (figure 33) contains the sprint goal, the list of objectives taken from the project backlog and the current time frame in which it is to be completed. Once completed a debrief document is made (figure 34).

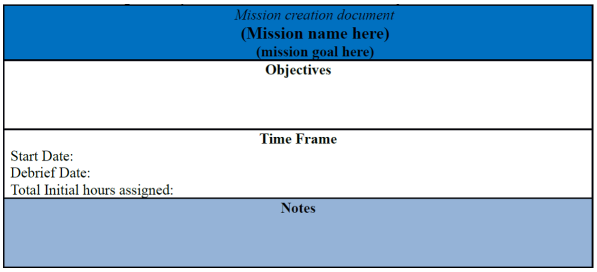

Figure 33: Mission creation document.

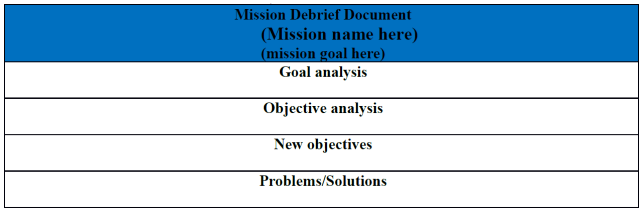

Figure 34: Mission debrief document.

UML Modelling UML or Unified Modelling Language will be used to model the structure of any application or computer program. UML contains a

number of standardised rules to present any software model allowing for anyone with knowledge of UML to critic and diagnose how the software will run and its internal modules are related.
Work flow The work flow design for the project is an amalgamation of XP SCRUM and DSDM practices. Traditional DSDM suggests that projects should be undertaken in a 7 step life cycle.

1. Pre-project.
2. Feasibility Study.
3. Business Study.
4. Functional model iteration.
5. System design + Build iteration.
6. Implementation.
7. Postproduction.

4.0.13 Game Description

The Coastal Defender game is envisioned as a puzzle game akin to Bust-a-move or Tetris where the player is trying to overcome environmental challenges as opposed to a specific antagonist. It also has aspects of Resource management, forcing the player to make decisions with heavy constraints.

It will be a turn based game set within a 2D hexagonal grid. At the end of each turn a series of rules will be run checking to see if the attributes of the hexes change.

The player has the ability to policy to each hex, affecting its behaviour. These policies mirror the Coastal Groups designations of Hold the Line, Advance the Line and Managed Realignment.

There is no specific win condition, it has been designed as is an exercise in decision making to produce the best result according to that player. However loose conditions do exist, a number of hexes will be marked as critical and if lost they player looses the game.

In addition there will be the function built into the game so the player can forward the current game state to social media for their peers to review.

4.0.14 Game Concepts

- **Hexagonal Grid:** Hexagonal grids are common in strategy games in particular a mainstay of hard-core table top war games. The advantage over square grids they enjoy are the six explicit relationships they enjoy with their neighbours, allowing for greater complexity to emerge. Hex grids can be designated according to the Cartesian coordinate system in a manner identical to a square grid.

- **Turn Based:** Turn based games are almost exclusively strategy or puzzle games, the turn based environment encourages deliberation within the player. Greater deliberation will hopefully foster a greater opinion on this subject.

- **Cell Designation:** At the beginning of each turn the player can designate the nature of any one hex changing its behaviour. These reflect the real world policy as designed by DEFRA and the coastal groups and allow the player to customise their game experience to their taste.

- **Cell Environments and Critical Locations:** The attributes of each cell will be dynamic with their environment being able to change from marine to terrestrial should the rules demand it. However a number of cells will be marked as critical (for example Tichwell Bird Marsh) and if the environment of these cells change it will trigger a loose state.

- **Resource Management:** In Coastal Defender the player is provided with resources in the form of money and the action of designating a cell will deplete this resource. The amount depleted will vary according to the type of designation and the modifier of the terrain type that cell currently has. At the beginning of each turn the resource will be refreshed.

4.0.15 Board Game Prototype

The board game prototype is a simple proof of concept for the coastal defender game mechanics. The rules are simple a number of hexagonal cards are laid down marked as either marine or terrestrial. For each turn the player rolls a dice for each terrestrial hex that has one or more marine neighbours. On a roll of 3 or less the cell is converted into a marine environment on a roll

of 4 or 5 it remains terrestrial and on a roll of 6 the adjacent marine cells are converted into terrestrial environments.

Figure 35: An example board layout

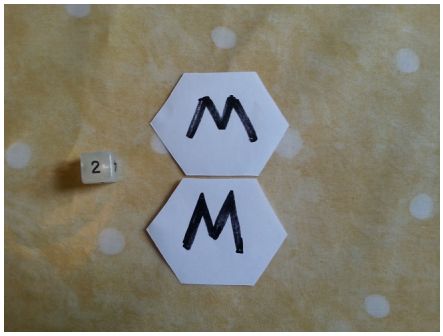

Figure 36: An example roll of 3 or less converting the environment into marine hex.

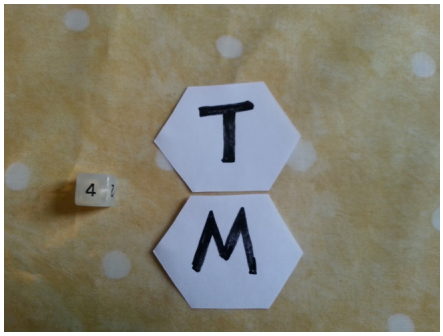

Figure 37: An example of a roll of 4 or 5 causing the hexes to retain their status-quo.

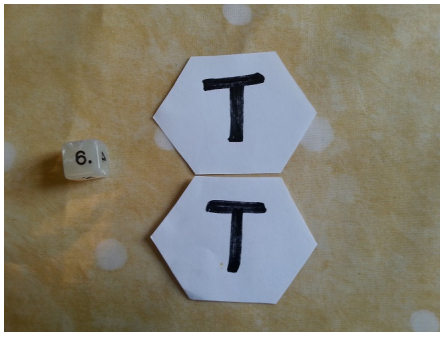

Figure 38: An example of a roll of 6, which converts the adjacent marine hex into a terrestrial hex.

4.1 Use Cases

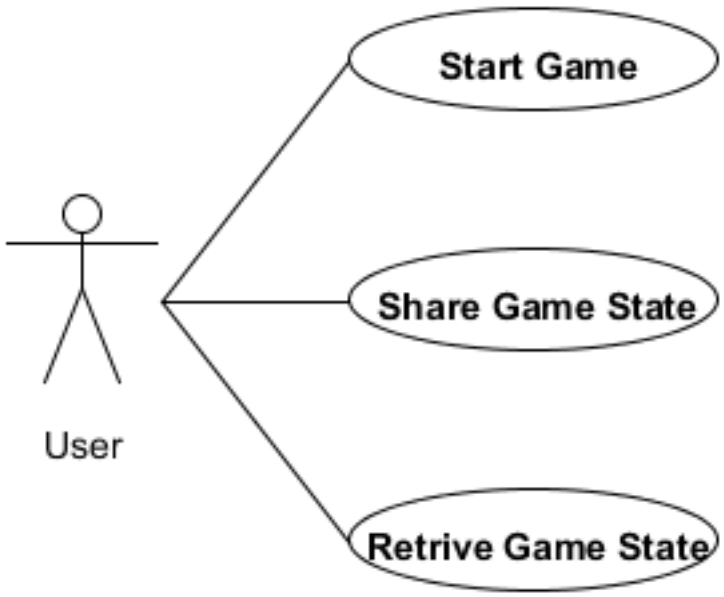

Figure 39: High level use case of the Coastal Defender Game.

4.1.1 Low-Level Use Case 1

Use-case: Start a new Game
Primary Actor: End User.
Goal: To run through a full game of twenty turns, allowing the user to share their game state at any point
Preconditions: Current access to the internet, both the server hosting Coastal Defender and the social media it links to.
Trigger: The user clicks the link to start a new game.
Scenario:
User: clicks the link to start a new game.
System: Loads initial game state.
User: May designate cells to modify their behaviour, or share the current game state.

System: updates current game view, or generates tag for social media status to be shared.
User: clicks next turn, allowing the same actions found in step two.
System: Runs algorithms to update game state.
Exceptions: User tries to designate an improper cell. System will respond with an error message explaining which cells are eligible for designation.

System is unable to update game state into new turn. Will raise an error informing the user.
Open Issues: How will the Ajax passing the game state to the server for augmentation work? What format will the data be passed to the server in ?
Use Case Diagram:

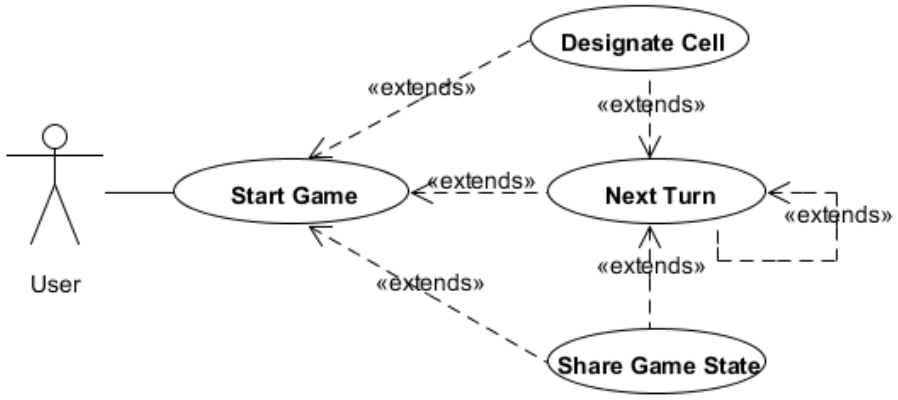

Figure 40:

4.1.2 Low-Level Use Case 2

Use-case: Share game state
Primary Actor: End User.
Goal: To update the user's social media status with a copy of the game state.
Preconditions: The user has access to the Coastal Defender game on the internet and an account to the social media platform chosen.
Trigger: User clicks the share game state button
Scenario:
User: clicks the share game state button.

System: Creates a format of the game state to be forwarded as a status on the social media.
Exceptions: The system is unable to create a new game state, an error will be raised to the user and attempt to resolve. The system is unable to forward the game state to the social media. The error raised will suggest checking social media's availability.
Open Issues: What will be the limitations of the social media be that will effect how the game state be presented? (for example if Twitter is chosen there is a 140 character limit)
Use Case Diagram:

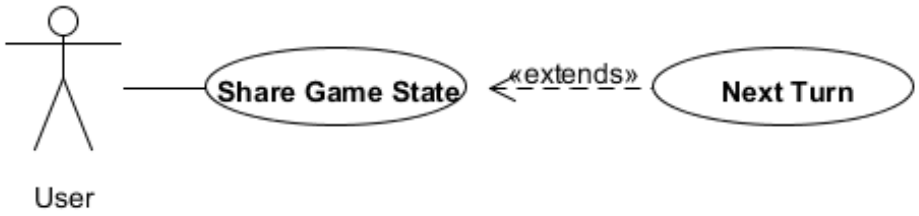

Figure 41:

4.1.3 Low-Level Use Case 3

Use-case: Retrieve game state
Primary Actor: End User.
Goal: The user to load up the current game state from a social media status.
Preconditions: For the user to have a social media account. For the user to have access to the Coastal Defender game on the internet.
Trigger: The user clicking on the link in the social media status.
Scenario:
User: Clicks on the social media status link.
System: Uses the unique game state label found in the social media status to generate the game state.
Exceptions: The system is unable to translate the label within the status into a new game state. An error will be raised

Open Issues: How will the link within the status appear? Which social media platform will be best suited for this system?
Use Case Diagram:

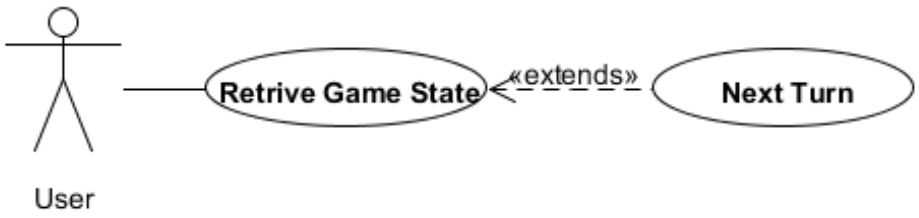

Figure 42:

5 Technology Overview

5.1 Google Maps Earth Browser

5.1.1 Overview

One of the central questions for the Coastaldefender project is to determine if Earth browser technologies are a suitable platform for this application type. The main advantage of using an Earth browser is that they are ubiquitous within the social framework of the internet. Coastaldefender will be using Google Maps, the reasoning for this is due to its overwhelming dominance on both web and mobile platforms. Additionally it has a well maintained API, with scores of libraries supplementing its functionality. This strong support from the developer community surrounding it will greatly assist in overcoming any potential technical issues during the development phase of this project.

5.1.2 Key Functionalities

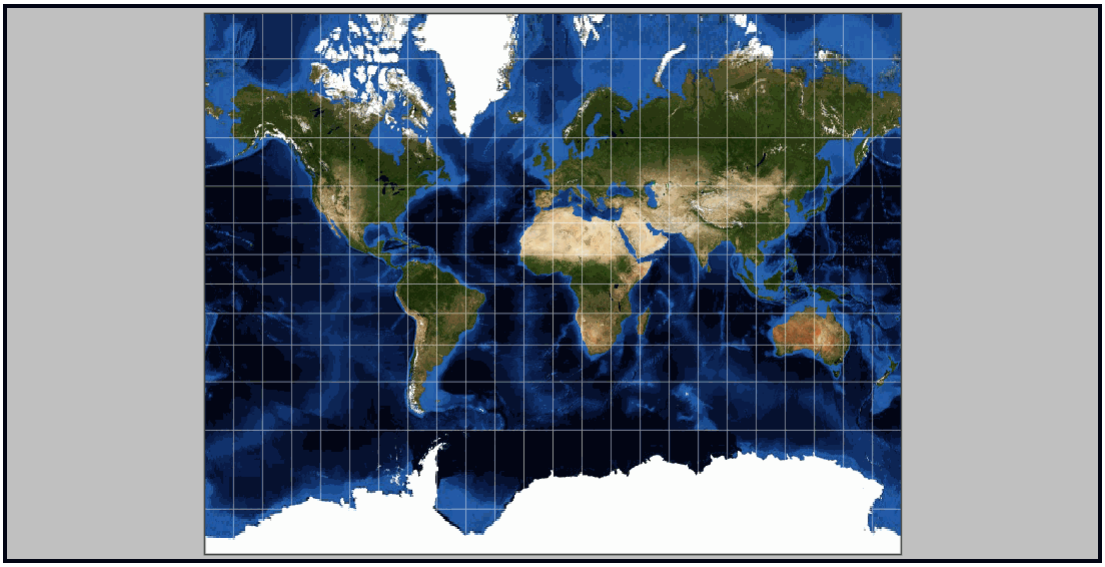

Figure 43: The Mercator projection as shown by NASA's projection series [66].

Google Maps displays the world within a Mercator projection, which is often considered to be the standard projection type due to its ability to retain the shape and navigation lines (such as longitude and latitude). However this projection type converts the world map into an cylindrical form and thus distorts the size of landmasses, with increased distortion size the further you travel from the equator [36].

Google Maps sees regular updates to its landmass imagery. Since it uses the imagery from Landsat there is a complete image archive of the Earth available every fourteen days. However in reality only specific portions of the world's surface will be updated at any one time, so that as stated in Google Map's very own help all images on Google Maps is approximately one to three years old [37].

It also should be noted that in the event of a large event such as a disaster or international sports event special attention and regular updates will be focused on that region [93]. Once a new update is made (approximately twice a month) a KML file is released showing all of the updated areas bordered with a red line.

The extensive API available for Google Maps provides huge amounts of customizability and extensibility to the product. The web interface API is JavaScript and makes plentiful use of DOM events to maximise the customizability. For example it has custom event listeners that will trigger on map zooms. Google Maps also supports KML (Keyhole Markup Language), which has become the de-facto markup language for geo-locational data. This allows for the content provider to display large amounts of data within the map through simple and human readable data.

Google Maps really excels at displaying large amounts of geo-locational data, be it through streamed CDTA data, native JavaScript or KML. These can appear in the form of simple data points, polygons, lines or markers displaying relevant information on a location through an info window [36].

5.2 The Django Web Framework

5.2.1 Overview

Django is described on the Django project homes page in the following terms. "Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design" [29].

It adheres to the DRY (Do not Repeat Yourself), where the developer is encouraged to reduce duplication and redundancy of code through its structure. The back end of Django is almost entirely Python, where data models are used to create a SQL database that form the backbone of the website. Another particular advantage of Django in the case of any geo-web application is that it supports PostGIS a geospatial PostgreSQL database.

5.2.2 Data Models and Views

Data models are python classes from which instances are used to populate the Django database. Through the Django admin the user can create these new instances, which in turn are mapped to a HTML template via a view function.

```
1 class textEntry (models.Model):
2     heading = models.CharField(max_length=50)
3     body = models.TextField()
4     order = models.IntegerField()
5
6     def __unicode__(self):
```

```
7         return self.heading
8         return self.body
```

In the example of the `textEntry` class the heading and body fields can be populated with text, whilst the order field can be used as a key to determine its position on a page. This is then served to the client side via a view function.

```
1 def startMenu(request):
2     textOrder = textEntry.objects.all().order_by('order')
3     t = loader.get_template('homeScreen.html')
4     c = Context({
5         'textOrder': textOrder,
6     })
7     return HttpResponse(t.render(c))
```

The `startMenu` view retrieves all of the instances of the `textEntry` class and sends them as part of the response object to be rendered on the webpage according to instructions by the template.

5.2.3 Templates

Templates also adhere to the DRY principle. A base HTML template is created which allows further HTML documents to extend from this using the Django template language. For example if a base HTML page contained the template tags {% block content %}{% endblock %} any template using the same tags will extend the base HTML with the content found within those tags.

```
1  {% block content %}
2    {% for textEntry in textOrder %}
3      {% if  textEntry.order == 1%}
4        <div class="textDiv">
5          <h2>
6            {{textEntry.heading|linebreaks}}
7          </h2>
8          <div>
9            {{textEntry.body|linebreaks}}
10         </div>
11     {% endif %}
12   {% endfor %}
13 {% endblock %}
```

The template here directly extends the base HTML and uses the class fields of the data model served by the view function as variables to place them onto the web page.

5.2.4 Admin access to the database

The admin site allows the user to interrogate, add, edit or delete any instance of an existing class model. Since these admin entries are represented within the website itself, it provides a quick and easy tool to update the data being viewed on the client side.

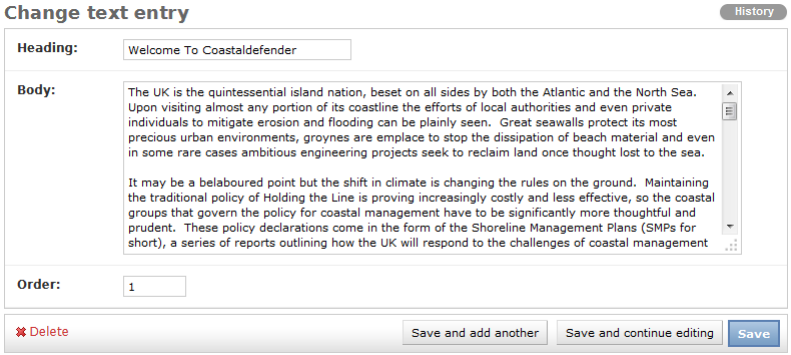

Figure 44: The Django admin form to add content for an instance of a data model class.

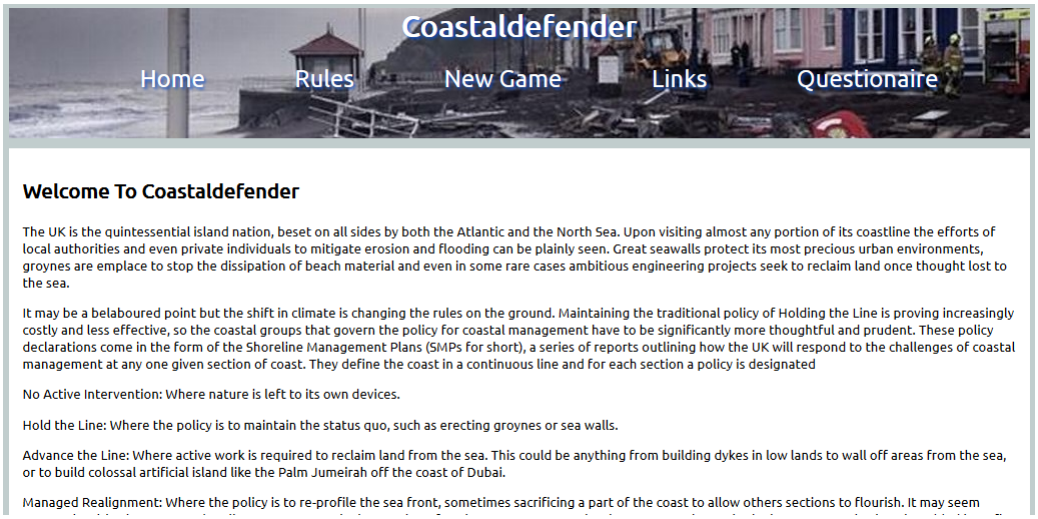

Figure 45: How Django renders an instance of a data model to the browser.

The content for the example of the `textEntry` data model to be displayed on the screen is shown in figure 44, when applied it will appear on a webpage in the manner shown in figure 45.

5.3 Languages and JSON

Coastaldefender uses the Django web platform, which limits the choices for scripting languages that can be used with it. On the client side the HTML is augmented with JavaScript (and its derivative JQuery), whilst on the server side it is exclusively Python.

5.3.1 JavaScript and JQuery

The Google Maps web API is written entirely in JavaScript thus making use of it as the main client side scripting language a necessity for this project. The JQuery JavaScript library is also used regularly throughout the client side scripts. JQuery simplifies many DOM interactions and streamlines POST and GET calls to the server.

5.3.2 Python

Since Python is the language Django is written in all server side functions are written in Djnago pattern Python.

5.3.3 JSON

JSON is used as the data transfer format for Coastaldefender. More light weight than XML, it also enjoys native support within Python and Django. This allows for conversion of packaged JSON sent to the server into its Python analogue and visa versa. KML was also discounted as a data transfer type due to its limitations to be generated dynamically.

5.4 Web Hosting

5.4.1 Hosting Service

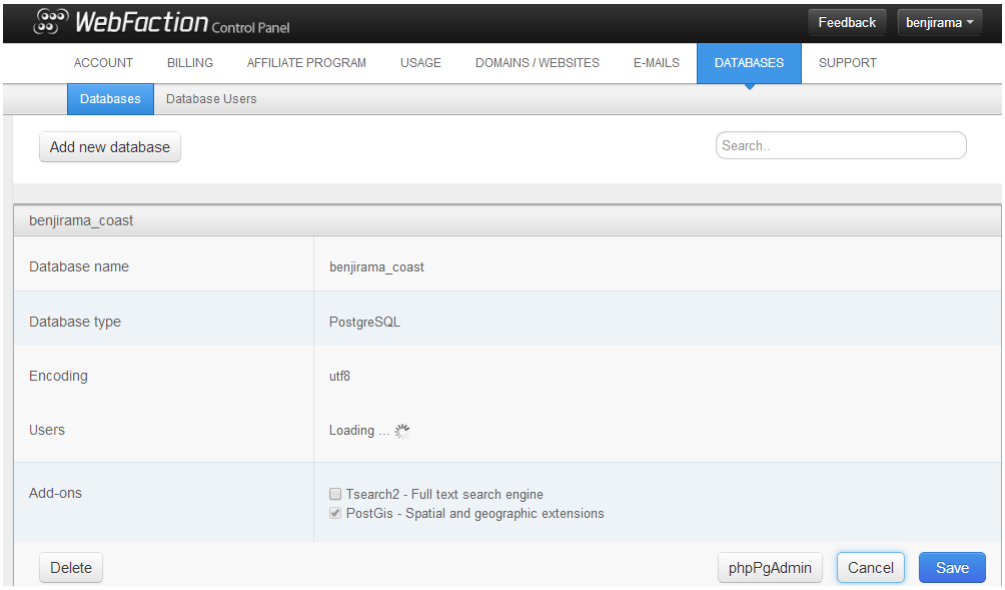

Figure 46: The Webfaction UI allowing high level editing of the user's web applications and easy access to PhpPgAdmin [105].

Coastaldefender is hosted via the Webfaction hosting service, an Apache2 based server solution. Webfaction provides strong customer support and

multiple development platform options, which Django. In addition it comes with an inbuilt UI to provide high level control of the development. Webfaction also provides access to PhpPgAdmin, a web tool used to provide extra control over the back end SQL database. This synergies well with any Django based project [105].

5.4.2 Development Tools

The development of Coastaldefender utilises FTP via the Filezilla open source FTP software [**?**] and SSH secure links to the Webfaction server through the Putty software [92]. All programming was done through notepad++ [**?**].

5.5 Social Media Integration

The Coastal Defender game at its very heart is a tool to be used by a community to generate discussion about environmental and geo-engineering issues. The best way to gain access to an online community for a project such as this would be through the integration of a social media platform through its respective API. The array of social media platforms available is staggering, but not all will be suitable for this project. In this section a feasibility study will be carried out of a number of social media platforms. The platforms considered are Facebook [25], Twitter [102], and Reddit [75]. Each represents a substantially different way for the communities to interact. The requirements for each platform will be considered against the following criteria.

5.5.1 Requirements

Accessibility:

The aim of this project to reach as a wide an audience as possible. The platform should be easy to use and have as a low barrier for entry as possible. This requirement will be judged by the platforms' ubiquitousness, its barrier for entry and ease of use.

Community Appropriate:

Different social media platforms attract different social groups. For example Pinterest has become a hub for arts and crafts enthusiasts, while the infamous site 4chan is notable for its highly active yet disruptive community; in both cases the communities for these sites would be inappropriate for the

project. Coastaldefender, seeks a community that is diverse, engaged and democratic.

Context Appropriate:

The platform should have representative aspects witin it that are suitable for the Coastaldefender project. For example since Instagram deals with the dissemination of photographs and thus would be inappropriate for integration.

API Useability:

The functionality of the API needs to be considered. In particular it is an absolute necessity that the Coastaldefender game can be linked to, commented upon and shared in the status. Ideally there will also be some way for the platform to tag various portions of the status with different meta tags (such as Twitters hashtag), this would allow for greater control over aspects of the game state transference between status update and game creation.

5.5.2 Facebook Study

Acessibilty:

+ Utterly ubiquitous in its usage. With over one billion users at the time of writing this, for many Facebook is the "front page of the internet".
+ The barrier for entry is low. Signing up is easy, and due to its huge user base, friends and family will often set up Facebook profiles who find the process difficult.
+ Easy to use.

Community Appropriate:

+ Due to the sheer number of users, Facebook has an extremely diverse population. It is often the case for individuals who have no other online presence to still have a Facebook account. The outcome being that Facebook is perhaps more representative of the general population than other social media platforms.
+ Facebook groups allow for a diversity of interests and subcultures.

Context Appropriate:

- General Facebook usage is social interactions amongst friends and family. Interests such as Costaldefender would be kept within the Facebook groups.

Finding a way to circumnavigate this could easily create a stream of status updates that could be considered as spam.

API Useability:

+ It is easy to share, comment upon and link to external data and media.
+ Massive status size limits. Facebooks status size limit is a huge 63,206 characters.
- No meta-tagging is available.

5.5.3 Twitter Study

Acessibilty:

+ Widely used. Not as nearly as ubiquitous as Facebook, but Twitters user base is still huge at approximately 500 million users.
+ The barrier for entry is low, and like Facebook the signing up process is easy.
+ Twitter is exceptionally easy to use. It is limited in its capabilities but powerful in its implementation.

Community Appropriate:

+ Diverse user base. Due to the scale of its user base Twitter has a vast array of diversity. The nature of its follow mechanism means that communities tend to focus around certain individuals. This may be celebrities, but equally SMEs (subject matter experts) in a particular niche.
- Unlike Facebook, Twitter is seen by some as a tool for the tech savvy only. Therefore Twitter perhaps does note have quite the real world diversity that can be found in Facebook.

Context Appropriate:

+ Due to the nature of communities building around SMEs Twitter tends to encourage the dissemination of technical information.
+ The brevity of Tweets, makes Twitter appear alot less obtrusive, and less likely to be considered spam (though it does exist on Twitter).

API Useability:

+ Sharing, commenting and linking to outside data and media is easy.
+ Twitters greatest advantage is its method of being able to metatag the tweet with its Hashtag system. Hashtagging and the @ system allows for

use within Coastaldefender to present the game state as a link or a unique hashtag that could be further commented upon.
- Obscured history. History on accounts tends to be partially obscured though not impossible to search through. This means that the Twitter discussions are nearly always current, but sometimes lack a historical perspective.
- The 140 character limit. Twitter has proven that 140 characters is a fine amount to generate discussion. However it will add technical difficulties whilst trying to transfer the game state back and forth between Twitter and the game.

5.5.4 Reddit Study

Accessibility:
+ Easy to sign up to and to use.
- Reddit has a significantly smaller community base than some of the other major platforms. For many people Reddit is considered a community of the tech savvy and thus they feel alienated from it.

Community Appropriate:
+ Reddit is known for its collaborative and active community, which takes up stories in the news and champions them. For example in 2012 a US bus driver (Karen Klein) was shown on CCTV footage being abused by passengers. Reddit spearheaded an online fundraiser which donated $500,000 to her retirement [75].
- Reddit is equally known for a portion of its community to be spiteful and disruptive. These Trolls are they are known exist on all social media, but due to Reddits pervasive internet culture they are particularly noticeable.

Context Appropriate:
+ Reddit covers topics of nearly all types, and there are well defined Subreddit forums where specific topics are discussed.
- Unless a persistent community is built round a certain topic it is quite easy for subjects to fade into obscurity. Reddit has a propensity to be subject to fads.

API Useability:
+ Sharing, commenting and linking to outside data and media is easy.

+ Status updates have a large upper character limit at a total of 10,000 characters.
- No metatagging ability.

5.5.5 Summary

Reddit, despite its extremely active and engaged population is too prone to drama and particularly malevolent trolls. The website even has a number of subreddits (sub forums) dedicated to recording the ongoing arguments within the community [76]. Facebook whilst having the broadest user base also lacks any meaningful way of tagging data. Outside of specialised groups, specific topics are rarely discussed due to its focus on the users personal life. Twitter on the other hand has the tremendous advantage of being able to tag metadata, to show the user which parts of the message is relevant. Its character limit which on the surface may seem hugely constricting appears to have neither slowed its discourse or popularity. Due to the nature of following individuals and searching for hash tags it is very easy to sort through the various tweets to find specific information. In balance it appears that Twitter will be the most suitable platform for use with Coastaldefender.

6 Game Implementation

6.1 High Level Overview

The following description of the site and its components use the Django web platform and thus its file/folder structure is in strict adherence to it. For greater detail on the development of the site and its components please refer to the appendix.

6.1.1 Site Overview Description

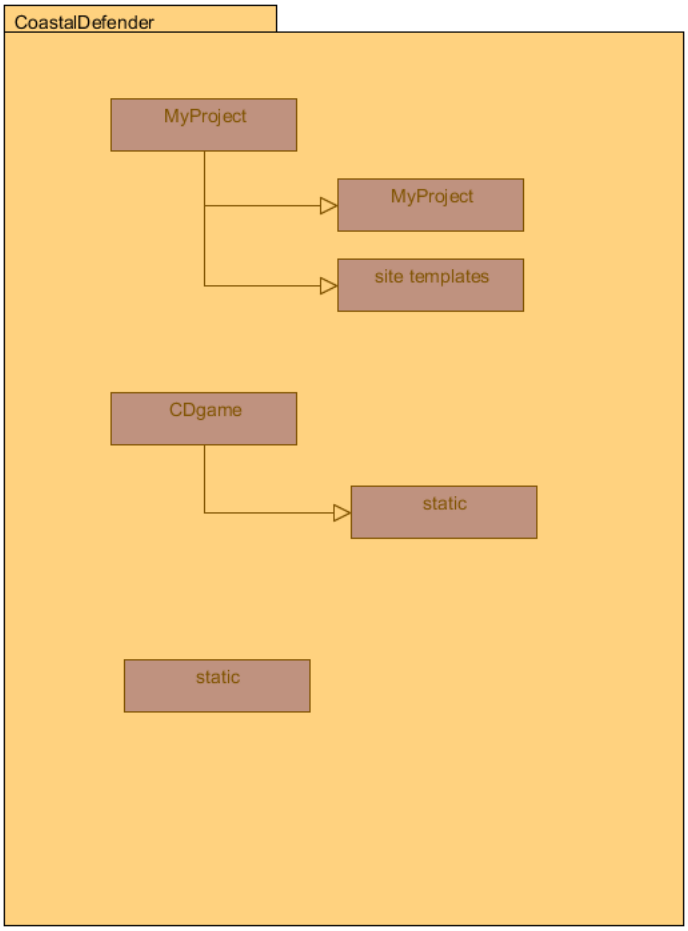

Figure 47: Package diagram for the overview of the Coastaldefender Django project.

The Django project folder holds three sub directories. MyProject which contains the main site settings, url redirects and HTML templates. CDgame is considered to be a Django app, where a group of models and views are grouped together as an application, the bulk of the CoastalDefender site data is stored here. The static directory is where Django has been explicitly told to retrieve all static files from. To resolve any confusion, the static folder

within CDgame is used to store static files specific to that app. During development, through Python server commands these static files are collected and are copied to the main static folder.

6.1.2 MyProject Description

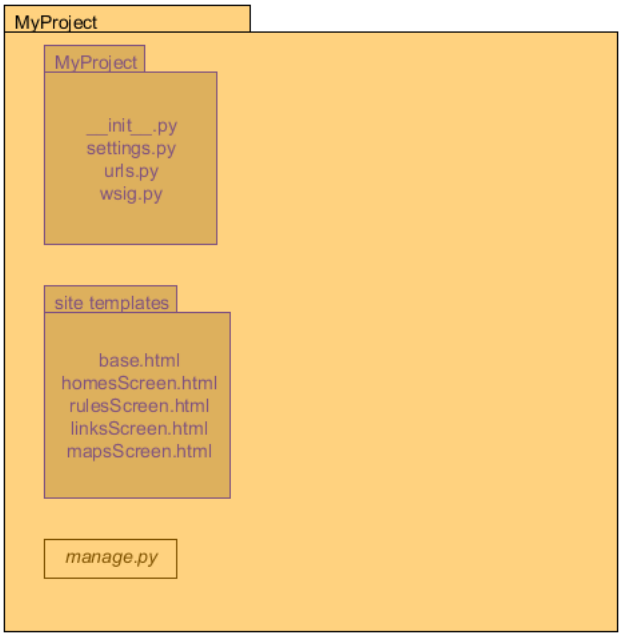

Figure 48: Package diagram describing the contents of the MyProject subdirectory.

The MyProject folder contains directories required for the site to be configured correctly. The manage.py file found in this directory contains commands that are used in development, such as collecting the static files and syncing the database with models found in Django apps.

The MyProject subdirectory contains four files. __init__.py and wsig.py are not touched through the development. Settings.py contains the site settings, such as the local time zone, language as well as the designated SQL database to be used (in this case postgresql). The root directory location is where all static and template files are to be stored to. Finally all apps

created for use within this Django project need to be registered within this file.

Settings.py

```
STATIC_ROOT = '/home/user/webapps/static'
STATIC_URL = 'http://user.webfactional.com/static/'
TEMPLATE_DIRS = (
    '/home/user/webapps/map/myproject/templates'
)
INSTALLED_APPS = (
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.sites',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'django.contrib.admin',
    'django.contrib.admindocs',
    'CDgame',
)
```

The urls.py controls url retrieval and redirects. In the case of urls suffixed with CDgame they are redirected to a local url.py file in the CDgame app directory.

urls.py

```
urlpatterns = patterns('',
    url(r'^admin/doc/', include('django.contrib.admindocs.
       urls')),
    url(r'^admin/', include(admin.site.urls)),
    url(r'^map/', include('CDgame.urls')),
)
```

The site templates folder contains all of the HTML templates that will be served for this project. The templates all extend from base.html, which is described below. The example of the `textEntry` in section 5.2.3 would extend directly from base.html. MapScreen.html is where the main content of the Coastaldefender game is found.

base.html

```
<html>
        <head>
                {% load staticfiles %}
                <link rel="icon" href="favIcon.gif" type="
                   image/gif" sizes="16x16">
```

```
5                   <script type="text/javascript" src="http://
                        code.jquery.com/jquery-1.8.0.min.js"></
                        script>
6                   <link href='http://fonts.googleapis.com/css?
                        family=Ubuntu:400,400italic' rel='
                        stylesheet' type='text/css'>
7                   <link rel="stylesheet" type="text/css" href="
                        {% static "coastStyle.css" %}">
8                   <script src="{% static "siteUtilities.js" %}"
                         type="text/javascript"></script>
9                   {% block extrahead%}
10                  {% endblock %}
11
12          </head>
13
14          <body>
15
16                  {% block nav %}
17                  <div id="navMenu">
18                          <h1>Coastaldefender</h1></br>
19                          <ul id="navBar" class="nav">
20                                  <li>{% block nav-home %}<a
                                        class="navOption" href="{%
                                         url CDgame.views.
                                        startMenu %}">Home</a>{%
                                        endblock %}</li>
21                                  <li>{% block nav-rules %}<a
                                        class="navOption" href="{%
                                         url CDgame.views.
                                        rulesMenu %}">Rules</a>{%
                                        endblock %}</li>
22                                  <li>{% block nav-game %}<a
                                        class="navOption" href="{%
                                         url CDgame.views.
                                        buildFrame %}">New Game</a
                                        >{% endblock %}</li>
23                                  <li>{% block nav-links %}<a
                                        class="navOption" href="{%
                                         url CDgame.views.
                                        linksMenu %}">Links</a>{%
                                        endblock %}</li>
24                                  <li>{% block nav-questions %}
                                        <a class="navOption" href=
                                        "{% url CDgame.views.
                                        questionMenu %}">
```

```
                                        Questionaire</a>{%
                                        endblock %}</li>
25                          </ul>
26                  </div>
27          {% endblock %}
28
29                  <div id='main'>
30                          {% block initial%}
31                          {% endblock %}
32                          {% block content %}
33                          {% endblock %}
34                  </div>
35          </body>
36          {% block controls %}
37          {% endblock %}
38  </html>
```

6.1.3 CDgame Description

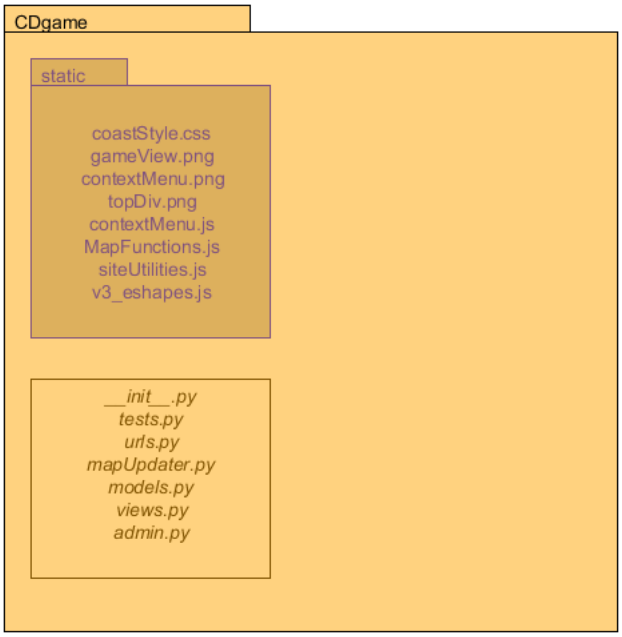

Figure 49: Package diagram describing the contents of the CDgame subdirectory.

The CDgame app folder contains the bulk of the code specific to CoastalDefender __init__.py and tests.py are again used in the Django back end and are left untouched during development. Urls.py calls any redirected url to its corresponding view for example. A url of ‘‘.../map/CDgame/buildFrame/’’ would be redirected from the MyProject urls.py to the CDgame urls.py which then calls the view `buildFrame`.

urls.py

```
urlpatterns = patterns('',
    url(r'^$', 'CDgame.views.startMenu'),
    url(r'^rulesMenu/$', 'CDgame .views.rulesMenu'),
    url(r'^questionMenu/$', 'CDgame .views.questionMenu'),
    url(r'^linksMenu/$', 'CDgame .views.linksMenu'),
    url(r'^buildFrame/$', 'CDgame .views.buildFrame', name ='
       createFrame'),
    url(r'^buildFrame/mapmaker/$', 'CDgame .views.mapmaker'),
    url(r'^buildFrame/newTurn/$', 'CDgame .views.newTurn'),
    url(r'^buildFrame/loadOldTurn/$', 'CDgame .views.
       loadOldTurn'),
    url(r'^buildFrame/loadSavedGame/$', 'CDgame .views.
       loadSavedGame'),
)
```

The `buildFrame` view is found in views.py. This file deals with rendering models to webpages as well as POST calls to the server.

views.py

```
@csrf_exempt
def buildFrame(request):
    mapView = mapFrame
    t = loader.get_template('mapScreen.html')
    c = Context({
        'mapView': mapView,
    })
    return HttpResponse(t.render(c))
```

In the case of `buildFrame` it collects an instance of the model and renders it into the mapScreen.html.

models.py

```
class mapFrame (models.Model):
    frame = models.CharField(max_length=20)
    def __unicode__(self):
        return self.frame
```

Models.py contains all of the models which are rendered to the templates via the views.

Admin.py contains all the registrations for models on the admin site. This allows interrogation of all the SQL entries via the admin UI.

MapUpdater.py contains the bulk of the game logic. All JSON sent server side is converted into its python analogues and then augmented through the game rules here such as the seaPower function.

The static directory contains all of the static files specific to this app. Such as the css and image assets. It also contains the v3_eshapes.js and contextMenu.js library which are used extensively within coastal defender. It also contains mapFunctions.js where the majority of the client side game code resides.

6.2 Use Cases and Gameplay

6.2.1 Game Elements and User Interface

Using the use cases from section 4.1.2 the following are gameplay examples of how a user would interact with the game. In each of the following use cases they will be interacting with the Coastaldefender user interface.

User Interface

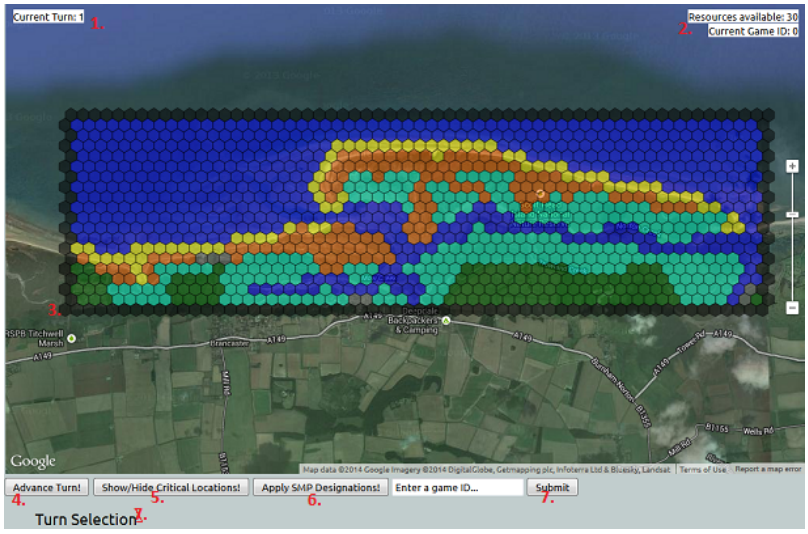

Figure 50: The Coastaldefender UI.

1. Current Turn Display: Shows the current turn in the game, there are ten turns in a game of Coastaldefender.

2. Resources and Game ID: The resources available will deplete as you designate each hex with a policy, once your resources reach zero you will no longer be able to apply policies to new hexes. At the beginning of each turn your resources will refresh to their total amount. The Game ID is your unique passage through the game, on a later date if you enter the game ID in to the submittal box it will revert back to this state and you will also be able to search through the history of that game.

3. The Game Board: By right clicking on an individual hex you will open up a menu where you can choose the designation, this will deplete your resources accordingly.

   Hold the Line: Costs 2 resources and makes the hex more resistant to erosion.
   Advance the line: Costs 4 resources and make the hex significantly more resistant to erosion to the point there is a chance you can reclaim land from adjacent marine hexes.
   Managed Realignment: Costs 1 resource and makes it more likely the sea will erode this land hex converting it into a marine environment.
   Different coloured hexes represent different environments.
   Blue: Marine hexes, you have no control over these.
   Yellow: Beach hexes, easily eroded.
   Orange: Dune hexes, easily eroded.
   Turquoise: Saltmarsh hexes, resistant to sea erosion.
   Green: Human rural hexes, susceptible to erosion if the sea reaches it.
   Grey: Human urban hexes, resistant to erosion.

4. The Advance Turn Button: Clicking on this button will move you onto the next turn.

5. Show/Hide Critical Locations Button: Will show you how many critical locations remain.

6. Apply SMP Designations Button: Will apply all of the line policy designations according to the official designations by the Coastal Group.

7. Submittal Box: By entering a game ID into this box you can replay an older version of a game already completed by a player.

8. Previous Turn Selection: Here you can select a previous turn to skip back to in order to try out a different strategy.

Optional Elements

Figure 51: The effect of toggling the hide/show critical button, all hexes considered critical show white.

During the game the player can use two optional tools, the first is the hide/show critical button and the second is the apply SMP. By clicking the hide/show critical button the player can toggle between hiding and showing which hexes are marked as critical by changing the colour of the critical hex to white and visa versa.

Figure 52: The effect of clicking the apply SMP button (all hexes with a strong green border have the HTL policy automatically designated.

The Apply SMP significantly changes the way the game plays. By clicking this the game will automatically designate policy lines to each hex in accordance to the North Norfolk SMP. This does not effect their resources but allows the player to interact with and change the official designations to their choosing.

6.2.2 Start New Game

Use Case Goal and Trigger Where the player seeks to start a new game of Coastaldefender and then following it through to its end state. During later development it became apparent that the original stated twenty turn game was too many. After approximately ten turns the player can already see the outcome of their decisions. A full twenty turn game would become unnecessarily complex and laborious. The length of turns has been adjusted accordingly.

Process

Figure 53: The Coastaldefender site menu.

Upon selecting the New Game option from the menu the player is transported to the game proper, where they begin on turn one with fifty resources, please note at this time the game ID will be zero.

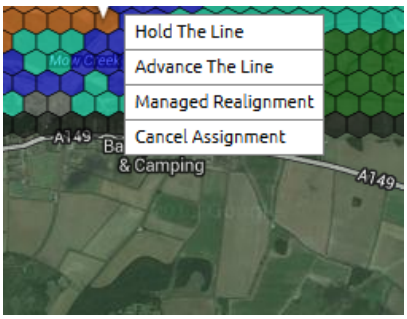

Figure 54: The context menu appearing after right clicking on a hex.

The player will start designating hexes with line policies by right clicking on the hex. This will bring up a context menu with the appropriate designation options as shown in figure 54. As the player applies designations the current resources UI object will decrease (two for HTL, four for ATL, and one for MR). If the player reaches zero resources they cannot place any further designations without removing existing ones. An example of the designations can be see in figure 55, where green is HTL, red is ATL and MR is purple.

Figure 55: A number of hexes designated with line policies.

Figure 56: The effect of game algorithms changing the nature of designated hexes.

The player then clicks the Advance Turn button and the outcome of the designations become apparent. An example of the outcomes from the previous designations in figure 55 can be seen in figure 56, in addition the game ID will generate a new number for this turn, the player repeats this process until the final turn is reached. **End State**

Upon reaching the tenth turn the player is informed that they have reached the end of the epoch and that they can summarise their attempts to protect critical locations by pressing the show critical locations button. The player is also then encouraged to share the game state via twitter.

6.2.3 Share Game State

Use Case Goal and Trigger

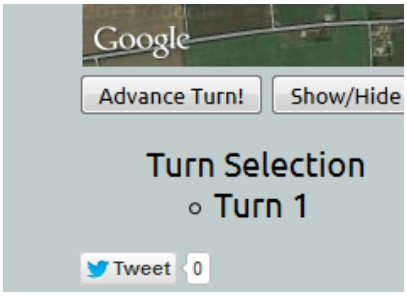

Figure 57: The embedded tweet button.

The player shares their current game ID through the Twitter social media platform. The trigger for this is simply clicking on the tweet button at the bottom of the game page. It is important to note that the player can tweet the current game state at any stage of the game. Coastaldefender generates a unique game ID for each unique history of game states.

Process

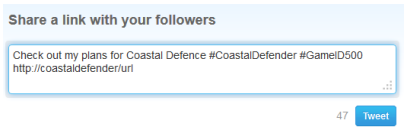

Figure 58: The text created by the game for sharing via twitter.

After clicking the tweet button the game will generate the text for a tweet with a #CoastalDefender tag and a #GameID tag unique to their game history as well as a link to the current Coastaldefender URL, an example of this is shown in figure 58.

End State

After amending the generated text if they wish to do so, the game state is then sent to Twitter as a tweet.

6.2.4 Retrieve Game State

Use Case Goal and Trigger Where the player seeks to load a game state from the Coastaldefender database. This is achieved by entering the game

ID into the Submittal Box. The game ID can be taken from a shared Tweet or other wise shared via a friend or colleague.
Process

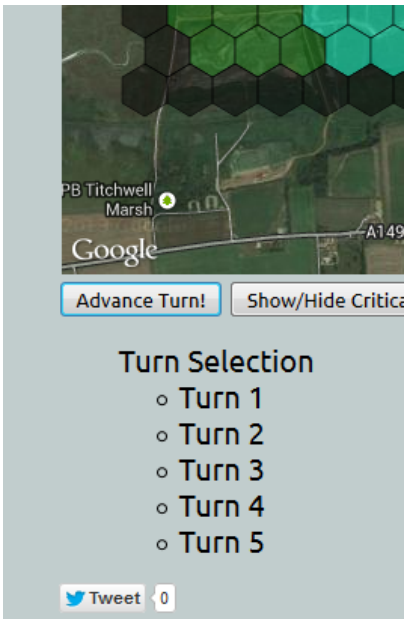

Figure 59: The list of previous turns the player may revert back to during their game.

Upon entering the game ID Coastaldefender will load the current turn as well as the complete history of turns for that game ID. This will result in a changed map layout and a list of previous turns. The player can then choose to revert back to these older turns by selecting them from the menu.
End State Once loaded the player may continue the game as usual.

6.3 Use Case State Diagrams

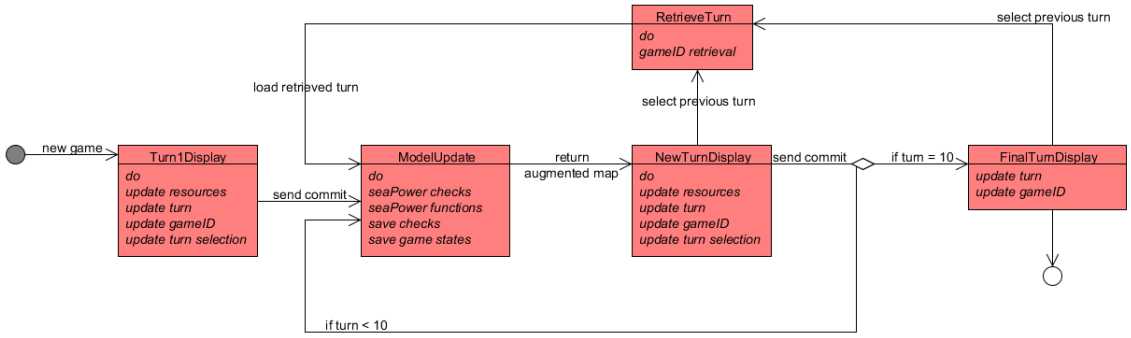

Figure 60: State diagram describing the state changes within Coastaldefender when implementing the Start a New Game use case.

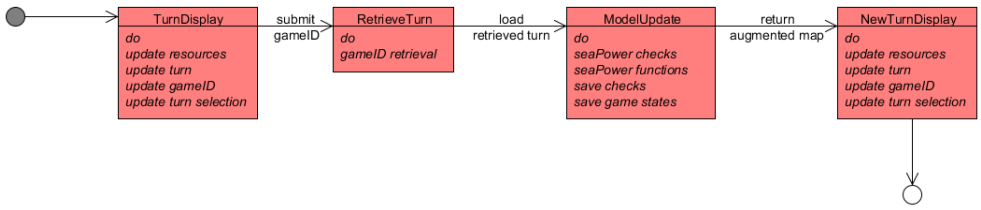

Figure 61: State diagram describing the state changes within Coastaldefender when implementing the Retrieve Game State use case.

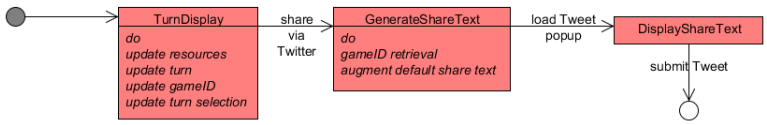

Figure 62: State diagram describing the state changes within Coastaldefender when implementing the Share Game State use case.

6.4 Use Case Sequence Diagrams

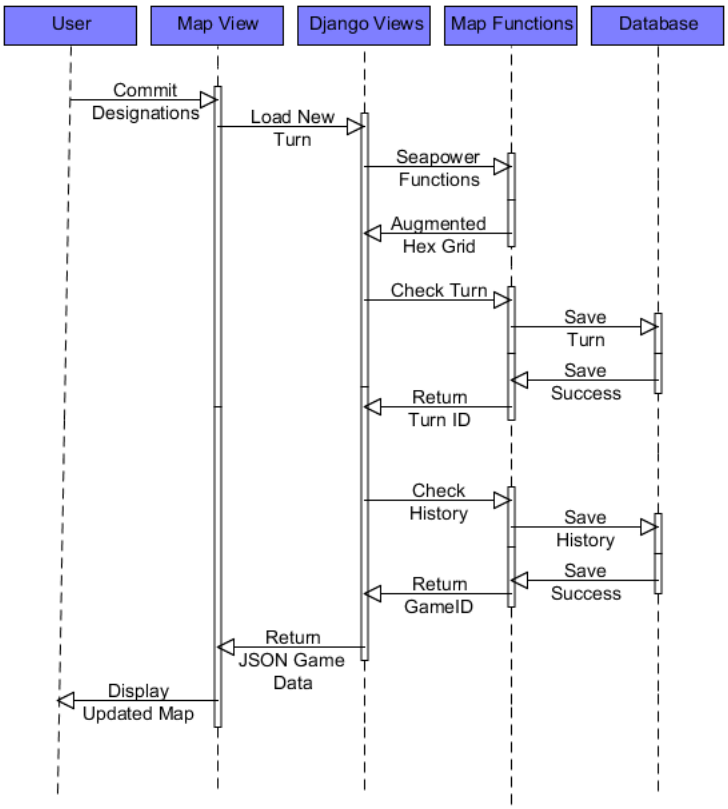

Figure 63: Sequence diagram describing the sequence of events in the Start New Game use case.

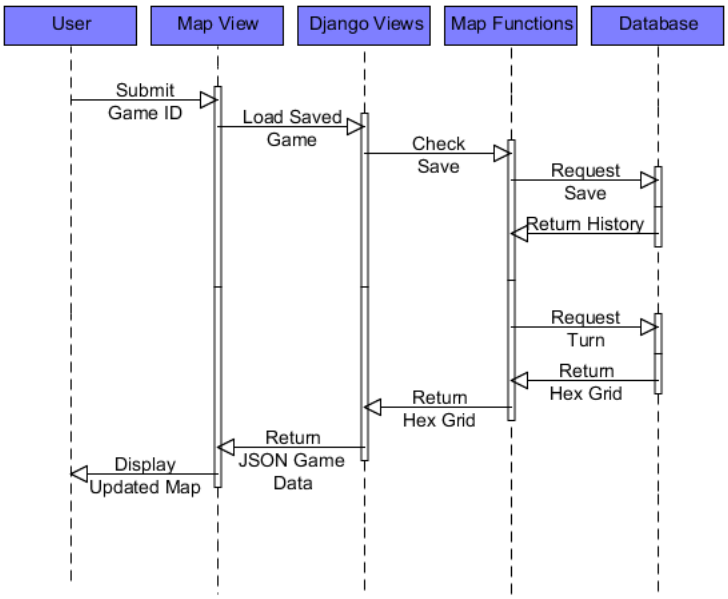

Figure 64: Sequence diagram describing the sequence of events in the Retrieve Game State use case.

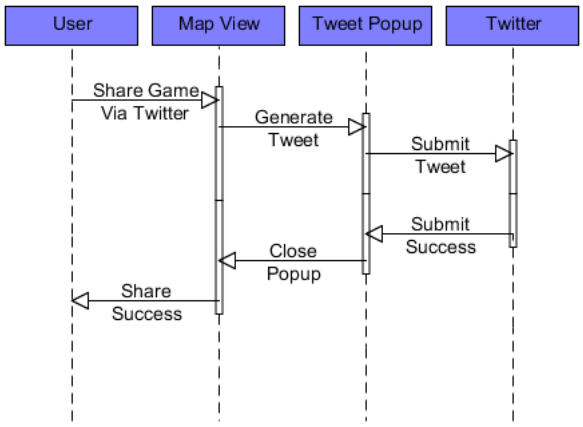

Figure 65: Sequence diagram describing the sequence of events in the Share Game State use case.

6.5 Component Diagram

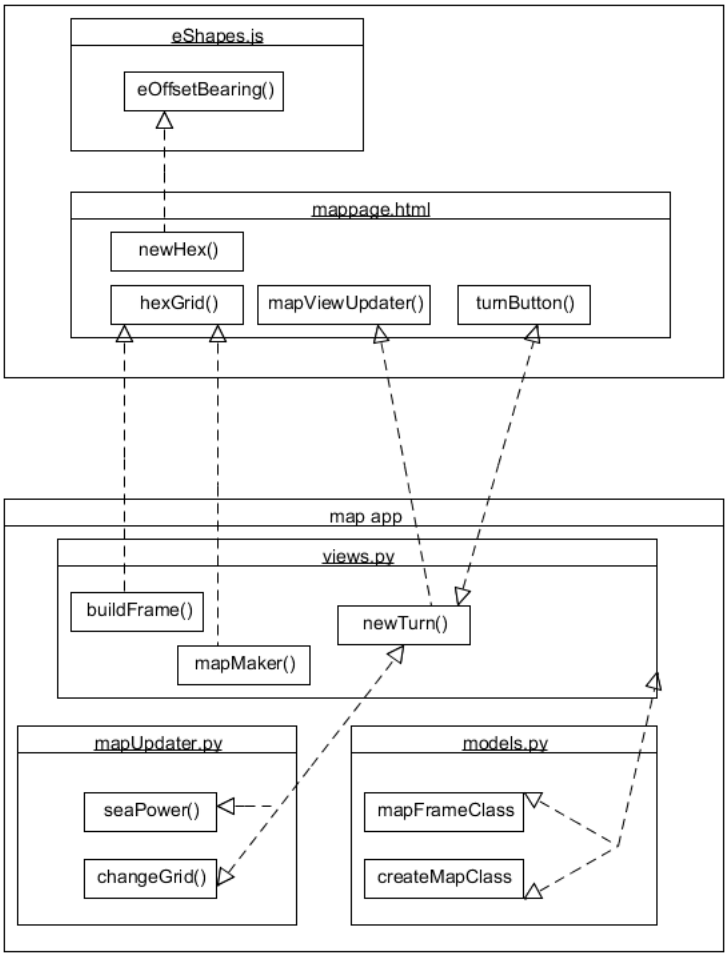

Figure 66: Component diagram showing the relationships between server and client side with the functions and objects that create these relationships.

6.6 Class Diagram

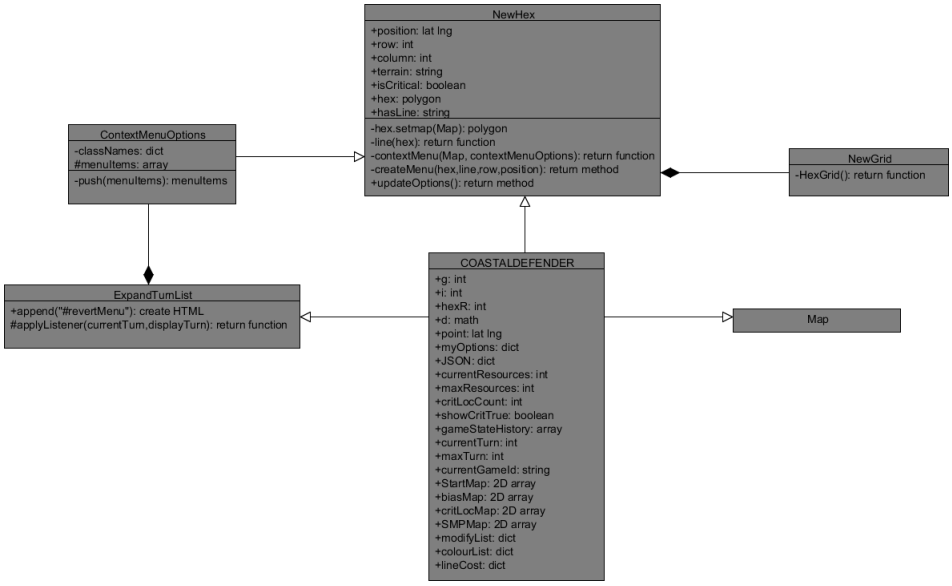

Figure 67: Class diagram describing the relationship between Coastaldefender's classes.

6.7 Low Level Description

6.7.1 Holder Object

The JavaScript Holder Object is a method championed by Douglas Crockford as a way to store persistent variables on the client side without having a cluttered list of global variables that can be prone to clashing. The `COASTALDEFENDER` global object can be amended to produce a varied number of maps and game behaviour.

models.py

```
1 var COASTALDEFENDER = {
2         g : 20,
3         i : 60,
4         hexR : 100,
5         d : 2*100*Math.cos(Math.PI/6),
6         point : new google.maps.LatLng(52.992367 , 0.604162),
7         myOptions : {
```

```
8                 zoom: 13,
9                 center: new google.maps.LatLng(52.976245 ,
                      0.67729),
10                mapTypeControl: false,
11                navigationControl: false,
12                zoomControl: true,
13                zoomControlOptions: {
14                        position: google.maps.ControlPosition
                              .RIGHT_CENTER
15                },
16                panControl: false,
17                streetViewControl: false,
18                mapTypeId: google.maps.MapTypeId.HYBRID
19        },
20        JSON : null,
21        currentResources : 50,
22        maxResources : 50,
23        critLocCount : 0,
24        showCritTrue : false,
25        gameStateHistory :[0],
26        currentTurn : 1,
27        maxTurn : 9,
28        currentGameId: "",
29        startMap : [],
30        biasMap : [],
31        critLocMap : [],
32        SMPMap : [],
33        modifyList : {
34        //The first half of this list are modifiers according
              to the terrain type of the hex.
35        'n' : 0, //null
36        'm' : 0, //marine
37        'b' : 10, //beach
38        'd' : 12, //dune
39        'sm' : 15, //saltmarsh
40         'hf' : 17, //human farmland
41        'hu' : 19, //human urban
42        //The second half of this list are modifiers
             according to the assigned line type to the hex.
43        'HTL' : 2, //hold the line
44        'ATL' : 4, //advance the line
45        'MR' : -2, //managed realignment
46        'NAI' : 0, //no active intervention
47        },
48
```

```
49          colourList : {
50          "nullHex" : "#000000",
51          "marineHex" : "#0000ff",
52          "beachHex" : "#ffff00",
53          "duneHex" : "#ff6600",
54          "saltmarshHex" : "#00ffcc",
55          "humanfarmHex" : "#006600",
56          "humanUrbanHex" : "#666666",
57          "HTLLine" : "#33FF00",
58          "ATLLine" : "#FF0000",
59          "MRLine" : "#660088",
60          "CritLoc" : "white",
61
62          },
63          lineCost : {
64          'HTL' : 2, //hold the line
65          'ATL' : 4, //advance the line
66          'MR' : 1, //managed realignment
67          },
68  }
```

Grid Drawing Attributes: The attributes of `g` (number of rows), `i` (number of columns), `d` and `hexR` (a measure of distance to be used for drawing hex polygons) are all used in the process of drawing the hex map on the grid. `Point` is the central latlng coordinate where the map centres upon. Amending these attributes can change the size of the grid or the hex polygons. The `myOptions` attribute is required for a Google Maps map instance to be loaded onto the page and contains options for the map controls and behaviour.

Game Asset Attributes: `JSON` whilst initially null is replaced with `JSON` which is temporarily stored here before being sent to the server. The `currentResources`, `maxResources`, `critLocCount`, `showCritTrue`, `currentTurn` and `maxTurn` attributes are all used in client side game logic to show the current turn, number of critical locations and resources available. The `gameStateHistory` and `currentGameId` attributes are used in the retrieval of previous game states should the user request it.

Map Attributes: The contents of `startMap`, `biasMap`, `critLocMap` and `SMPMap` have been removed for the sake of brevity. Each is a 2D array sharing the dimensions described in the `g` and `i` attributes. The `startMap` contains the initial terrain layout for the game. Each additional map mirrors `startMap`. `BiasMap` effects the server side `seaPower` functions, the

`critLocMap` denotes which hexes are considered critical. Whilst `SMPMap` contains the official SMP designations should the user choose to instate them.
Game Values Attributes: `ModifyList` describes the initial resistance a terrain type has when applied against the `seaPower` function, as well as the augmented value of applying a policy to that hex. `ColourList` contains the colours which are applied to hexes according to their terrain type and policy designation. `LineCost` is the cost in resources that a specific policy designation uses up.

6.7.2 Initial Setup

When the url "`.../map/CDgame/`" entered it is linked to the specific Django view as stated in urls.py, in this case the `buildFrame` view in views.py.
views.py

```
@csrf_exempt
def buildFrame(request):
    mapView = mapFrame
    t = loader.get_template('mapScreen.html')
    c = Context({
        'mapView': mapView,
    })
    return HttpResponse(t.render(c))
```

The `buildFrame` view renders an instance of the `mapFrame` model into the mapScreen.html. Once mapScreen.html has loaded the global variable of `newGrid`, calls the constructor of `HexGrid` which in turn calls the function of `buildGrid`.
MapScreen.html

```
newGrid = new HexGrid();
MapFunctions.js
function HexGrid () {
        this.grid = buildGrid();
}
function buildGrid(){
        var grid = new Array(COASTALDEFENDER.g);
        gridPoint = COASTALDEFENDER.point;
        for(var x = 0; x<COASTALDEFENDER.g;x++){
                grid[x] = new Array(COASTALDEFENDER.i);
                gridPoint = displacement(x,gridPoint);
                for(var y = 0; y<COASTALDEFENDER.i;y++){
```

```
13                          var gridHex = new NewHex(x,y,
                                gridPoint);
14                          grid[x][y] = gridHex;
15                          grid[x][y].createMenu(grid[x][y].hex,
                                grid[x][y].hasLine,x,y);
16                          if(COASTALDEFENDER.critLocMap[x][y]
                                == 1){
17                                  COASTALDEFENDER.critLocCount
                                        = COASTALDEFENDER.
                                        critLocCount + 1;
18                          }
19                  }
20          }
21          return grid;
22  }
```

The `buildGrid` function uses a 2D for loop to create a 2D array of `NewHex` objects. Please note that the `buildGrid` function uses the `g` and `i` attributes of the holder object to determine the size of the hex grid. Additionally with each instance of `NewHex` it also calls the `createMenu` prototype function to create the context menu associated with that hex. It uses the `displacement` function to place the new row correctly in correlation to the previous.

views.py

```
1   function NewHex(xCord,yCord,gridPoint){
2           this.position = EOffsetBearing(gridPoint,
                COASTALDEFENDER.d*yCord,90);
3           this.row = xCord;
4           this.column = yCord;
5           this.terrain = null;
6           this.isCritical = null;
7           this.hex = new google.maps.Polygon.RegularPoly(
                EOffsetBearing(gridPoint,COASTALDEFENDER.d*yCord
                ,90),COASTALDEFENDER.hexR,6,0,"#000000",1,1,"#00
                ffff",0.5,true);
8           this.hex.setMap(map);
9           this.hasLine = "NAI";
10          this.line = google.maps.event.addListener(this.hex, '
                rightclick', function(mouseEvent){
11                  buildLine(xCord,yCord);
12          });
13          this.contextMenu = new ContextMenu(map,
                contextMenuOptions);
14  }
```

TheNewHex constructor creates a new instance with the following attributes.
Position: its latLng position on the map.
Row: its x Cartesian coordinate.
Column: its y Cartesian coordinate.
Terrain: currently null but will contain the terrain type.
IsCritical: currenly null but its value is determined by the critLocMap.
Hex: an instance of the hexagonal polygon drawn onto the map via the V3_eshapes.js library.
HasLine: its current designation type NAI means no active intervention.
Line: creates a listener to create a context menu.
ContextMenu: links the context menu to the context menu options.

Once created a JQuery POST function calls the Django mapmaker view sending the current StartMap to the server.

mapScreen.html

```
1 $(document).ready(function(hexGrid){
2         jsonMap =       JSON.stringify(COASTALDEFENDER.
              startMap);
3         $.post("mapmaker/",{"startMap":jsonMap,"xLength":
              COASTALDEFENDER.g,"yLength":COASTALDEFENDER.i},
              function(data) {
4                   mapViewCreator(data);
5                   $(".twitter-share-button").attr("data-text",
                        "first turn");
6         }, "json");
7 });
```

The mapmaker view calls the buildGrid function which creates and returns a JSON object that mirrors the StartMap that can applied to the NewGrid, NewHex objects.

View.py

```
1 @csrf_exempt
2 def mapmaker(request):
3     startMap = json.loads(request.POST["startMap"])
4     x = json.loads(request.POST["xLength"])
5     y = json.loads(request.POST["yLength"])
6     rawMap = mapUpdater.buildMap(x,y,startMap)
7     return HttpResponse(rawMap)
```

mapUpdater.py

```
def buildMap(xAxis,yAxis,startMap):
    for x in range(0,xAxis):
        theGrid["hexGrid"].append([])
        for y in range(0,yAxis):
            theGrid["hexGrid"][x].append({
                "xCord":x,
                "yCord":y,
                "terrain":startMap[x][y],
                "modified":'',
                "hasLine":'NAI'
                })
    jsonGrid = json.dumps(theGrid)
    return jsonGrid
```

The returned JSON object is then compared against the `NewGrid`, `NewHex` objects and the values of the JSON object are applied to the grid. The `updateOptions` method is called on each hex to apply to appropriate colour to the `NewHex` object's terrain attribute.

MapFunctions.js

```
function mapViewCreator(data){
        COASTALDEFENDER.JSON = data;
        jsonGrid =      JSON.stringify(COASTALDEFENDER.JSON);
        COASTALDEFENDER.JSON = jsonGrid;
        for(var x = 0; x<COASTALDEFENDER.g;x++){
                for(var y = 0; y<COASTALDEFENDER.i;y++){
                        newGrid.grid[x][y].terrain = data.
                           hexGrid[x][y].terrain;
                        newGrid.grid[x][y].isCritical =
                           COASTALDEFENDER.critLocMap[x][y];
                        newGrid.grid[x][y].updateOptions();
                        }
                }
        }
```

6.7.3 Context Menu

Applying the context menu to each hex was a vital function allowing for policy designation. The context menu in this case is applied via the contextMenu.js library. It uses a set of options designated by a global object and an instance of context menu is applied to each individual hex.

MapScreen.html

```
1  var contextMenuOptions = {};
2  contextMenuOptions.classNames = {menu:'context_menu',
       menuSeparator:'context_menu_separator'};
3
4  var menuItems = [];
5  menuItems.push({className:'context_menu_item', eventName:'
       hold_the_line', label:'Hold The Line'});
6  menuItems.push({});
7  menuItems.push({className:'context_menu_item', eventName:'
       advance_the_line', label:'Advance The Line'});
8  menuItems.push({});
9  menuItems.push({className:'context_menu_item', eventName:'
       managed_realignment', label:'Managed Realignment'});
10 menuItems.push({});
11 menuItems.push({className:'context_menu_item', eventName:'
       cancel_assignment', label:'Cancel Assignment'});
12 contextMenuOptions.menuItems = menuItems;
```

The logic in the `CreateMenu` method is used to apply different designation types and apply colours accordingly.

MapFunctions.js

```
1  NewHex.prototype.createMenu = function (hexShape,lineDetails,
       x,y){
2        this.menuList = google.maps.event.addListener(this.
             contextMenu, 'menu_item_selected', function(latLng
             , eventName){
3             if(lineDetails == "NAI"){
4                  if (eventName == "hold_the_line"){
5                       if(COASTALDEFENDER.
                            currentResources -
                            COASTALDEFENDER.lineCost.
                            HTL <= 0){
6                            alert("You do not
                                 have enough
                                 resources to
                                 assign this policy
                                  to the hex.");
7                       }
8                       else{
9                            lineDetails = "HTL";
10                           newGrid.grid[x][y].
                                 hasLine =
                                 lineDetails;
11                           COASTALDEFENDER.
```

```
                                        currentResources =
                                         COASTALDEFENDER.
                                        currentResources -
                                         COASTALDEFENDER.
                                        lineCost.HTL;
12                                  hexShape.setOptions({
                                        strokeColor :
                                        COASTALDEFENDER.
                                        colourList.HTLLine
                                        , strokeWeight :
                                        6});
13                                  document.
                                        getElementById("
                                        over_map_right").
                                        innerHTML = "
                                        Resources
                                        available:" +
                                        COASTALDEFENDER.
                                        currentResources;
14                          }
15                  }
16
17                  if (eventName == "advance_the_line"){
18                          if(COASTALDEFENDER.
                                currentResources -
                                COASTALDEFENDER.lineCost.
                                HTL <= 0){
19                                  alert("You do not
                                        have enough
                                        resources to
                                        assign this policy
                                         to the hex.");
20                          }
21                          else{
22                          lineDetails = "ATL";
23                          newGrid.grid[x][y].hasLine =
                                lineDetails;
24                          COASTALDEFENDER.
                                currentResources =
                                COASTALDEFENDER.
                                currentResources -
                                COASTALDEFENDER.lineCost.
                                ATL;
25                          hexShape.setOptions({
                                strokeColor :
```

```
                                    COASTALDEFENDER.colourList
                                    .ATLLine, strokeWeight :
                                    6});
26                                  document.getElementById("
                                    over_map_right").innerHTML
                                     = "Resources available:"
                                    + COASTALDEFENDER.
                                    currentResources;
27                                  }
28                          }
29
30                          if (eventName == "managed_realignment
                            "){
31                                  if(COASTALDEFENDER.
                                    currentResources -
                                    COASTALDEFENDER.lineCost.
                                    HTL <= 0){
32                                       alert("You do not
                                            have enough
                                            resources to
                                            assign this policy
                                             to the hex.");
33                                  }
34                                  else{
35                                  lineDetails = "MR";
36                                  newGrid.grid[x][y].hasLine =
                                    lineDetails;
37                                  COASTALDEFENDER.
                                    currentResources =
                                    COASTALDEFENDER.
                                    currentResources -
                                    COASTALDEFENDER.lineCost.
                                    MR;
38                                  hexShape.setOptions({
                                    strokeColor :
                                    COASTALDEFENDER.colourList
                                    .MRLine, strokeWeight :
                                    6});
39                                  document.getElementById("
                                    over_map_right").innerHTML
                                     = "Resources available:"
                                    + COASTALDEFENDER.
                                    currentResources;
40                                  }
41                          }
```

```
42                  }
43                  else{
44                          if (eventName == "managed_realignment
                            " || eventName == "
                            advance_the_line" || eventName ==
                            "hold_the_line"){
45                                  alert("This hex has already
                                    been assigned.");
46                          }
47                          if(eventName == "cancel_assignment"){
48                                  lineType = newGrid.grid[x][y
                                    ].hasLine;
49                                  if (lineType = "HTL"){
50                                          lineValue =
                                            COASTALDEFENDER.
                                            lineCost.HTL;
51                                  }
52                                  if (lineType = "ATL"){
53                                          lineValue =
                                            COASTALDEFENDER.
                                            lineCost.ATL;
54                                  }
55                                  if (lineType = "MR"){
56                                          lineValue =
                                            COASTALDEFENDER.
                                            lineCost.MR;
57                                  }
58                                  COASTALDEFENDER.
                                    currentResources =
                                    COASTALDEFENDER.
                                    currentResources +
                                    lineValue;
59                                  lineDetails = "NAI";
60                                  newGrid.grid[x][y].hasLine =
                                    lineDetails;
61                                  hexShape.setOptions({
                                    strokeColor :
                                    COASTALDEFENDER.colourList
                                    .nullHex, strokeWeight :
                                    0.5});
62                                  document.getElementById("
                                    over_map_right").innerHTML
                                     = "Resources available:"
                                    + COASTALDEFENDER.
                                    currentResources;
```

```
63                              }
64                      }
65              });
66  }
```

7 Testing and Critical Analysis

7.1 Testing Methodology

7.1.1 Overview

The relative success of Coastaldefender is dependent on its ability to engage the pubic, therefore a practical test of the game must be carried out with active participants. Considering previous examples of game play or online crowd sourcing there seems to be some precedent for success using this model.

The aims of the testing will be to have a number of people to play Coastaldefender, then to measure their attitudes towards Coastaldefender via a questionnaire. The questionnaire will focus on Costaldefender's suitability as a tool and the user attitude towards scientific issues.

7.1.2 The Questionnaire

The questionnaire comprises of a number of positive statements to which the user can choose to their stance towards it. In these cases they can choose to: Strongly Agree, Agree, Neither Agree or Disagree, Disagree or Strongly Disagree. The questions are as below.
GameID: The Game ID of the game they have just played so the questionnaire can be linked to a game.
CoastalDefender is a game: To gauge how the user views Coastaldefender. Do they see it as a game or more as a tool?
Google Maps or a similar Earth browser (like Bing maps) is a sutible platform for this kind of application: To gauge the user's opinion in help to answer the first aim of this project.
CoastalDefender has made me more inclined to learn about coastal management and flooding: To gauge if it has an effect on engagement.
I often share information and opinions about about scientific and environmental issues on my social media networks: To understand

how people currently use social media in relations to environmental and scientific issues.
Communities affected by coastal flooding can use a tool like CoastalDefender to assist with engagement on a given issue: To gauge their opinion on its ability to engage.
Did you share information about the game through the Tweet button or other social media?: A yes or no answer, it is useful to see how people in practice.

The Questionnaire also has two questions capturing age range and gender. This is to provide some demographic context.

7.1.3 Crowd Sourcing Approach

Natural Social Network: To use existing social networks of family and friends to use Coastaldefender. Whilst crowd sourcing through this approach will provide the most naturalistic responses from a broad demographic it could possibly have the lowest level of engagement since there is no specific interest or incentive to use Coastaldefender.
Constructed Social Network: To build up an a presence of the Coastaldefender "brand" on social media and to create a community around it to provide testing towards. Anyone choosing to engage with Coastaldefender would be highly motivated due to their interest. Building an online presence can be a difficult, not to mention a time consuming task in itself. There is no guarantee that building a community will work in the first place.
Crowd Sourcing Platform: A small number of crowd sourcing platforms exist to perform micro-tasks. Amazon's Mechanical Turk [4] is perhaps the most well known example of these. Within these platforms the requester will create a pre-set number of HITs (Human Intelligence Tasks) and provide these to the free market of workers who will complete these tasks for a small financial incentive. This is perhaps the best way to source the largest number of responses but may lack quality responses in the face of a rush for cash.

7.2 Testing Implementation

7.2.1 Crowd Sourcing Issues

Initially the Coastaldefender project was going to depend on a crowd sourcing platform solution to find users for the game and complete the questionnaire.

The existing crowd sourcing platforms though few in number can attract a large number of workers in a very small space of time. Plus there is a novelty value to crowd sourcing an academic project in this manner. As stated in section 7.1.3 the financial incentive may mean that the results may not be sincere but for sheer volume it seems like the best approach.

The Mechanical Turk was chosen as the ideal platform for this, since it is by far the most established and most popular. However during the registration procedure it became apparent that only US citizens may be work requesters. The supposed reason for this is found in the controversial Patriot Act of 2001 [15] as allowing foreign money to flow through a US based service could aid money laundering for terrorism [47].

Thankfully a number of similar platforms exist which are not US based. Most are relatively small in size since this industry so new, and most are limited to the type of HITs a worker can perform. Of these alternative platforms Clickworker stands out as Mechanical Turk's main competitor [34]. Clickworker is a German based micro-task platform that tests individual users competencies and assigns them tasks according to their skills. However after registering with the site as a requester and completing the form for an order to be created, no communication was ever returned. Despite trying to contact Clickworker on a number of occasions through their internal contact form, no response was ever forthcoming.

With the testing frustrated, the project turned the existing social network platforms to promote the game and seek users.

7.2.2 Results

After approximately a week of disseminating Coastaldefender in existing social networks uptake was disappointing, with two hundred and forty two unique game IDs were created. Assuming each game produced ten turns that would be approximately twenty four individual games played. Yet only eighteen questionnaires were completed.

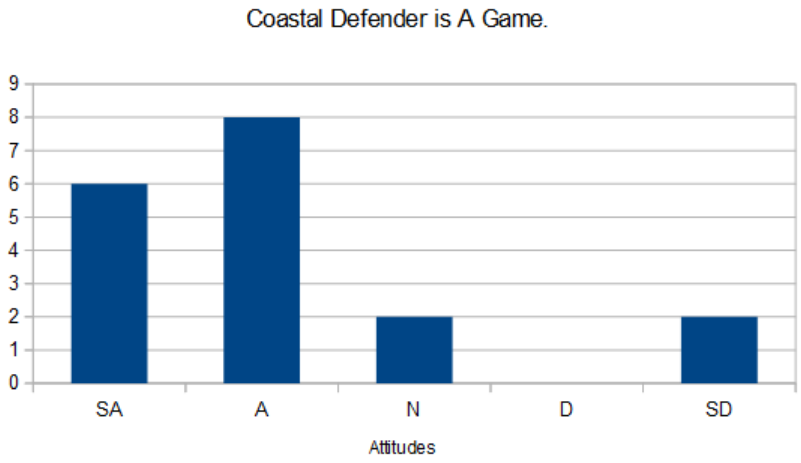

Figure 68: Results for Question 1.

Question 1: The overwhelming response for this question that they do indeed view Coastaldefender as a game.

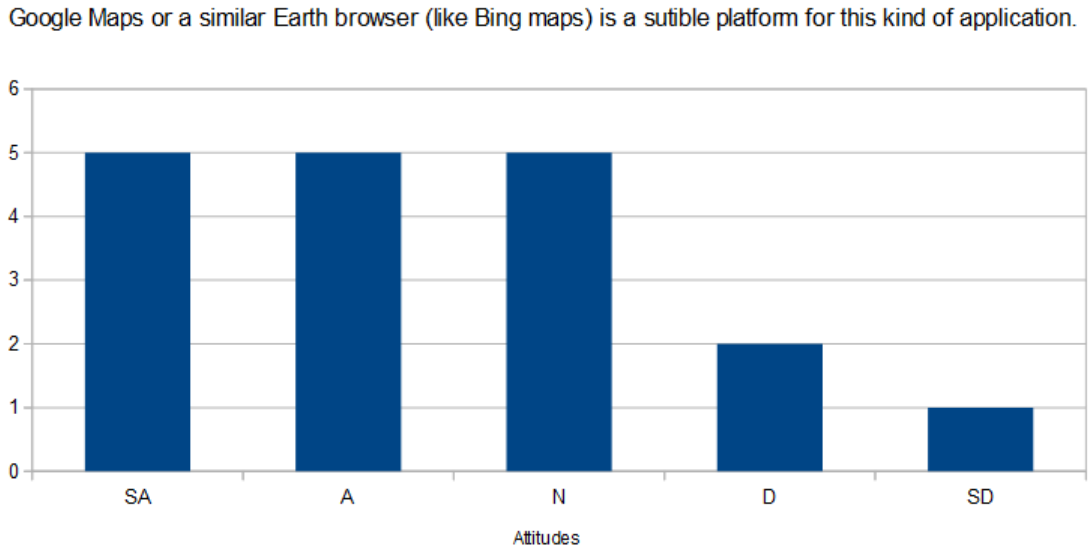

Figure 69: Results for Question 2.

Question 2: In response to question two there seems to be a mixed reaction

to question of using Earth browser for this kind of application, though there is an obvious positive skewing of the answers.

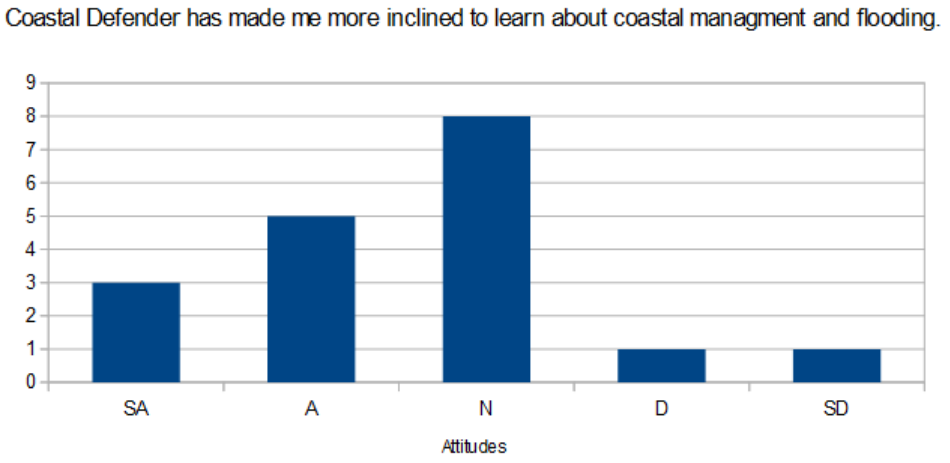

Figure 70: Results for Question 3.

Question 3: Despite a positive skew of the answers the vast majority of the users remain unmoved by Coastaldefender's attempts to engage them further in the topic of coastal management and flooding.

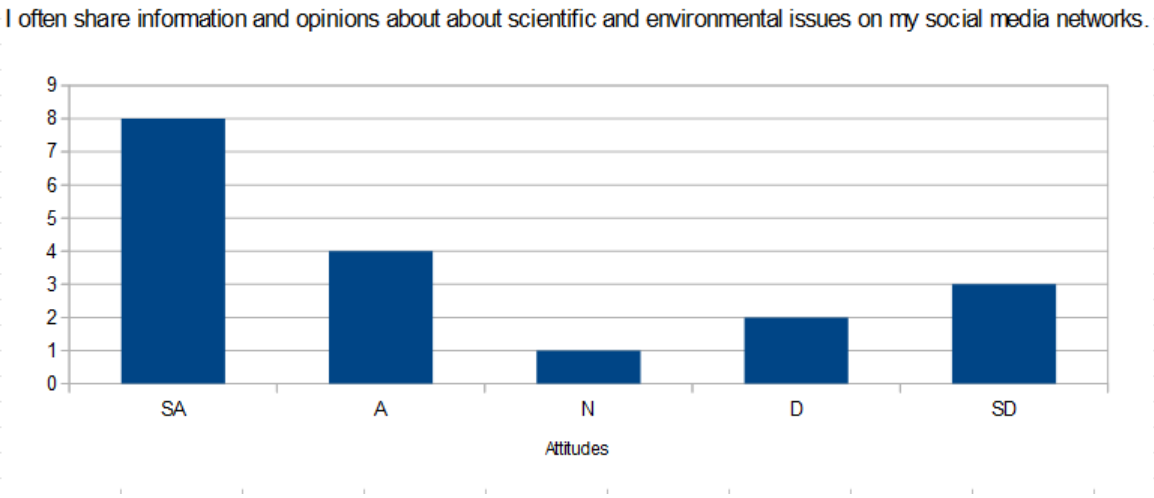

Figure 71: Results for Question 4.

Question 4: It appears that the users are likely to share information about scientific and environmental issues within social networks, though there is also a negative skewing for this question too. Of all the answers this creates the most prominent divide in users, and perhaps represents a difference in attitudes to those who use social media for personal reasons and those who use additionally use it as a source of news and information?

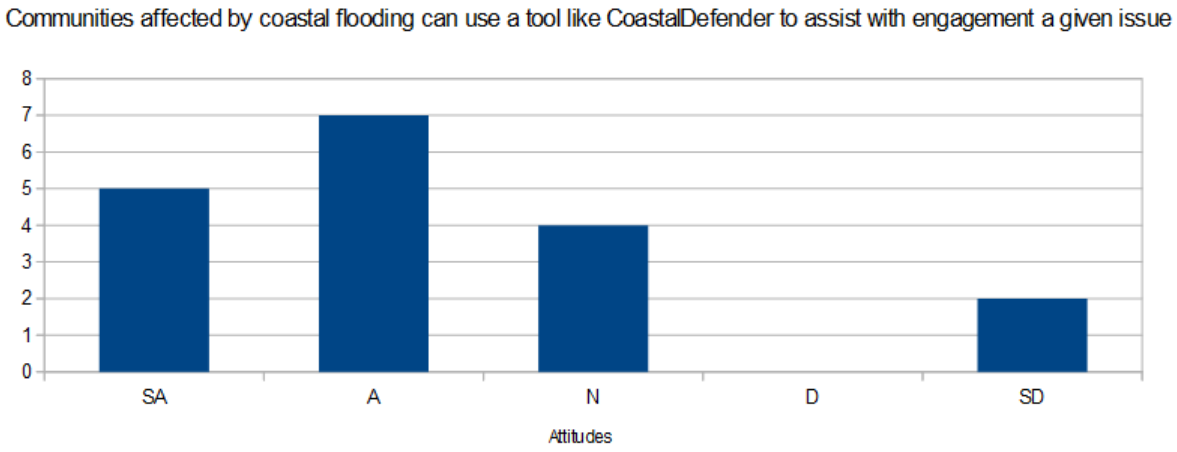

Figure 72: Results for Question 5.

Question 5: A strong positive skewing of the answers suggest that most people believe in some way Coastaldefender or a similar tool could be used to help communities facing coastal flooding issues.

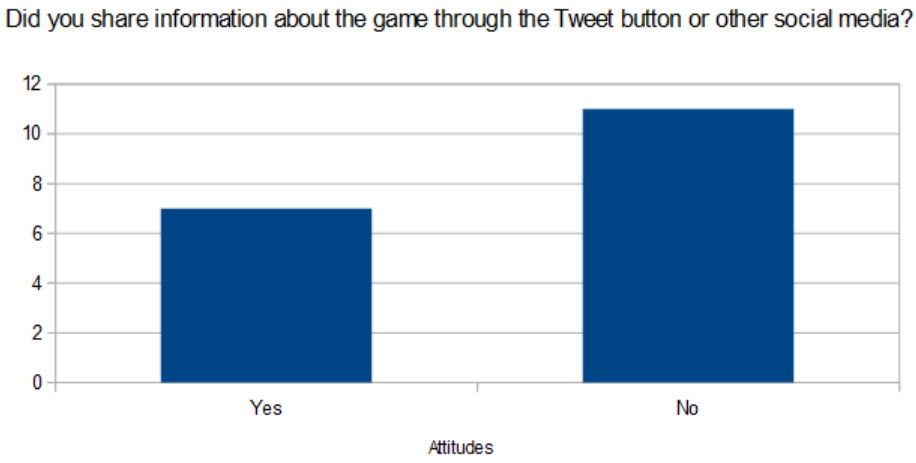

Figure 73: Results for Question 6.

Question 6: The majority of users did not share the information about Coastaldefender on social networks.

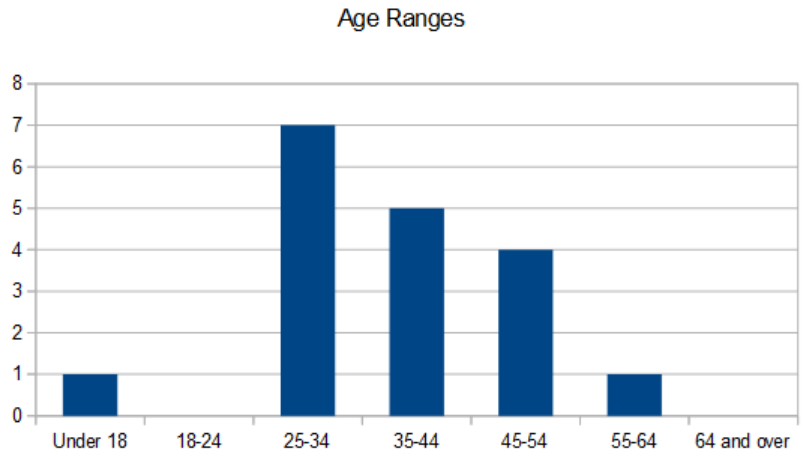

Figure 74: Results for Question 7.

Question 7: The first of the demographics showing that the majority of users for this project tend to be younger adults between 25-34 and this trends downwards as users age upwards.

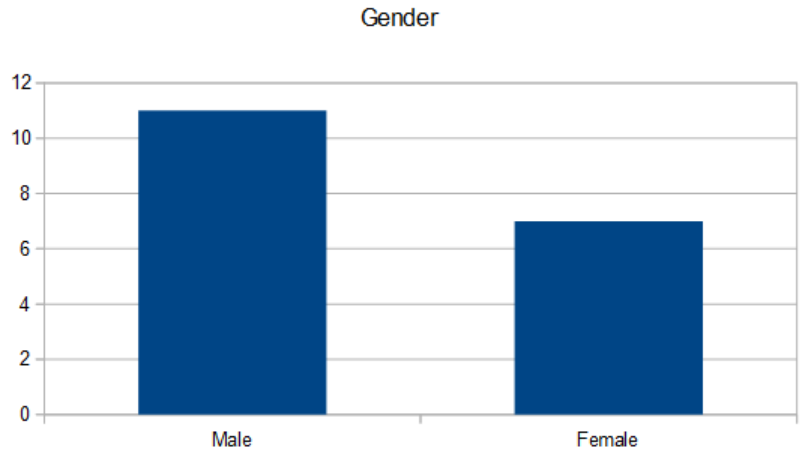

Figure 75: Results for Question 8.

Question 8: There is a much greater showing for male users than female.

7.2.3 Results Criticism and Analysis

The users who have answered the questionnaire are taken directly from the social network of the creator of Coastaldefender. Therefore they are more likely to share similar social and political views held by the creator. Since Coastaldefender was undertaken with view that technocratic solutions are a viable method to resolve real world problems, it is likely that the users views reflect this stance. Indeed there appears to be so validation of creator's view in the users answers. This may be a bias of the social group that the creator belongs to, or typical beliefs held in a larger population.

If it is the case that users from the creator's social network are more likely to mirror a technocratic view, then regardless of the sample size of users taken the results would have been largely similar.

The value of Coastaldefender: Users recognise Coastaldefener as a game and state that it is a suitable tool to be used within an Earth browser. Despite this most users feel that they are no more of less likely to be engaged with the issues of coastal management. Perhaps the reason for this is the limitations of the game, a more smooth and fun experience on a different platform may have a more positive outcome. Alternatively if the API was more supportive of these kind of applications a more engaging game could be created. However since the direction of the API seems to be focused on

displaying data as opposed to data manipulation the former solution seems like the more feasible path for greater success.

Coastaldefender and affected communities: There appears to be a strong opinion that Coastaldefender or an application similar to it can help local communities affected by coastal flooding to become engaged with the issue. This is in juxtaposition to the answers of question three where most respondents feel they personally are unaffected by Coastaldefender. One has to question then that if a member of a community faced with these issue was introduced to Coastaldefender that their opinion of the issue remain as unchanged as before like other game users.

Existing Engagement with scientific and scientific issues: Users are already strongly engaged with scientific issues. This may mean that the largely neutral response to question three might well be a moot point. The users may already be likely to investigate and share information about these issues readily with incentive required.

Females and Social Network Sharing: An interesting pattern does emerge, of the seven people who did share information on Coastaldefender four were female. Considering that the majority of users in the questionnaire were male it shows that when playing a game of Coastaldefender women are much more likely to share it with others on their social networks. This may well be a quirk of the users who partook in this questionnaire, or could be indicative of larger pattern. this is a pattern that could warrant further study.

7.3 SWOT Analysis

7.3.1 Overview

A SWOT analysis is a commonly used business tool to measure the current standing of a business model or product. SWOT is an acronym for Strengths, Weaknesses, Opportunities and Threats. This analysis will be comparing the success of Coastaldefender in light of the three stated aims of this project.

1. To determine if an Earth browser technology is suitable for developing a game like tool.

2. To determine if a game like tool is good for developing discussion around a specific environmental or scientific issue.

3. To determine if there is a novel or specific crowd sourcing model that can be implemented to achieve the second aim.

Since this section looks in detail at Coastaldefender's technology and implementation there will be less focus on aims two and three in this case.

7.3.2 Strengths

Strengths are abilities or traits which are a boon to Coastaldefender in terms of achieving its aims.

- **Web based technology:** The utilities Coastaldefender utilises are already fully available for the user. This means that no additional requirements or technological knowledge is required on their part, thus giving it a very low barrier for entry. Coastaldefender can be embedded in any website that uses a Django web platform due to the manner in which Django manages its app system.
- **Highly flexible:** Coastaldefender currently is set up to recreate the Scolthead region of the North Norfolk UK coastline. This was achieved by storing all the variables in the `COASTALDEFENDER` holder object. The maps which determines the terrain layout, the positions of critical locations and the inherent bias are all attributes of this holder object. By amending these and a number a Coastaldefeneder scenario could be set up for any section of UK coastline. In other words this could be used as a multi-site tool to be adapted and used where required.

7.3.3 Weaknesses

Weaknesses are abilities or traits that are considered to be detrimental to Coastaldefender achieving its aims.

- **Platform inappropriate:** The Earth browser technology is clearly not ideal for this kind of tool. Whilst excellent at displaying a large amount of data points and geo-locational information it appears to have more difficulty showing large amounts of graphical overlays. Usually these overlays are limited but in the case of Coastaldefender each hex is a separate shape and requires server calls to create and augment it. The result is creating something that is slower than required and

occasionally clunky to use. This is largely unavoidable, re-factoring the code may go some way to mitigate the issues but due to in which the grid is drawn (the hence all the objects within it) the growth of the algorithms is largely quadratic.

- **Redundant functionality:** Large amounts of the functionality specific to Earth browsers are not used within Coastaldefender. additionally a similar tool to Coastaldfender could have been created through a variety of other means. Its positioning on the map does allow for the context of the game to be understood, the importance of which should not be understated. Another application could recreate something similar to Coastaldefender.

7.3.4 Opportunities

Opportunities are external factors that could be co-opted to help Coastaldefender achieve its aims.

- **Unity Game Engine:** During the development stage of Coastaldefender Unity game engine 4 was released [95], which has gone on to massive success. Unity is a free game engine that can use either a Python like language or JavaScript as a scripting language. Large amounts of it is an object orientated system and can be managed with within its UI, making it fast and flexible. Unity could be an excellent platform into which Coastaldefender could be migrated, although it would require an additional download on the user's behalf. It could make a much more streamline and usable experience for the player, however the sacrifice would be the loss of map context.

7.3.5 Threats

Threats are external factors that could be that are detrimental to Coastaldefender achieving its aims.

- **Technology renting:** Coastaldefender is at the mercy of Google's usage policy and whilst there is no pattern in Google's history to suggest that an academic project using its technology would be subject to forced take down there is always that possibility if it is consider to misuse its product. In addition any changes in future iterations

of Google Maps could require significant re-factoring on the part of Coastaldefender if it changes fundamental aspects of the API.

7.3.6 Summary

Objectively as a tool Coastaldefender does provide the outcome of using Earth browser technology to create an interactive game like application. However the limitations of Earth browser technology to display large amounts of graphical data is evident. This creates Coastaldefender's largest flaw which is that interactivity is slow and clunky. Whilst the mechanics seem to work as intended they are hampered by the ability to create a seamless environment for the user. Additionally since the functionality of Coastaldefedner could be recreated within a number of different platforms. One has to consider that perhaps a more optimal non Earth browser solution could be used.

A more suitable alternative platform could potentially be the Unity game engine. Unity has tremendous support in both terms of its development team who provide online tuition and classes as well as an extensive developer community. The engine is well optimised and the number server calls it would have to make (as in the case of Google Maps) would be lessened considerably, therefore over coming Coastalderfender's greatest flaw. However this comes at a price which is two fold. Firstly for Unity web plugins to work additional downloads are required from the user thus raising the barrier to entry. Secondly it loses the ability to have the map context making portability from one UK coast location to another more difficult.

Overall Coataldefenders strength is its ability to use widely supported technology within the existing web browser framework. For non technical users accessing it this will be as simple as clicking on the web link. However the user experience would be much greater within a different platform.

7.4 Grand Summary

Coastaldefender is a successful project but only to a point. It pushes the ability of the Google Maps API well outside of its normal boundaries and its functionality suffers for it. Its slow and clunky nature is largely due to the Ajax calls.

There does appear to be some positive support from users of Coastaldefender to be used as a tool for communities affected by coastal flooding issues. With some work Coastaldefender could be embedded in a number of

different websites and could provide a useful addition to an educational or governmental site to engage their audience.

However perhaps a better solution would be to create Coastaldefender on a different platform which can create a far more fun and engaging experience. Unity 4 seems to be a prime target for such an undertaking, since the process of building applications on Unity is fast and enjoys strong developer support.

The largest disappointment for the entire project was the supposed panacea of crowd sourcing. It was assumed that using a micro-task platform such as Mechanical Turk would produce a quick turn around of a large amount of testers. The industry unfortunately appears to be still immature, with Mechanical Turk being the only serious platform. However for anyone outside of the US to enjoy access to its workers will require policy change in the US government. So just like the future of coastal management policy in the UK, the final decision is in the hands of politicians. Unless pressured by a motivated and informed public.

8 Appendix and Development Notes

8.1 Overview

The following documents are the complete development notes for the Coastaldefender project. Each project cycle represents a mission and its associated objectives.

8.2 1st Cycle

8.2.1 Overview

The overall goal of the 1st mission cycle is to create a template hex grid that will act as the structure through which the game mechanics will work. This will act as one part of a model where Google Maps and social networking functions will take place. The envisioned conceptual model for this project was named “the floating pyramid”.

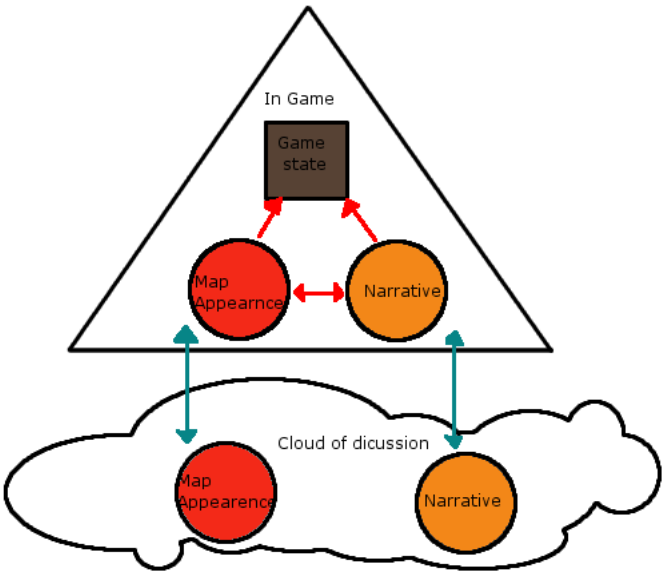

Figure 76: The Floating Pyramid Model

The pyramid model outlines the three necessary features for the game.

1. **Map Appearance:** The game view. This is what the user will interact with to play the game.

2. **The Game State:** This will contain the saved game states that will be interchangeable within the Map Appearance. Changing the chosen game state will change the Map Appearance.

3. **The Narrative:** The narrative of the game is determined by the decisions of the player(s), and thus affects the hex grid accordingly to these choices. The narrative is also the shared information about the game within the user's social media networks.

When creating the hex grid the following requirements need to be considered.

1. **To be variable in size and shape**: The hex grid will be based on Cartesian coordinate based grid so it will always be quadrilateral. Since the area in to which the game can apply could vary controls need to

be introduced that will be able to change the size and shape of the hex grid.

2. **For each individual hex to be an object:** The values in each hex grid will be both persistent but also mutable. Thus having each individual hex as an object will be an absolute requirement.

3. **To create a grid so that each hex can be identified by Cartesian coordinate:** This allows for mapping within the game and also allows individual players to interrogate the information stored within a hex object.

8.2.2 Objectives

This project has three primary objectives and one secondary objective.

1. **Discover Method:** Find a way to create a hex grid on Google maps

2. **Create the hex grid:** Use the method to create the hex grid of a predetermined size.

3. **Implement Cartesian Coordinates:** Find a way to apply a unique Cartesian coordinate to each hex within the grid.

4. **Vary the grid:** A secondary objective. Find a function that will change the dimensions of the hex grid.

8.2.3 Designing the Hexagonal Grid

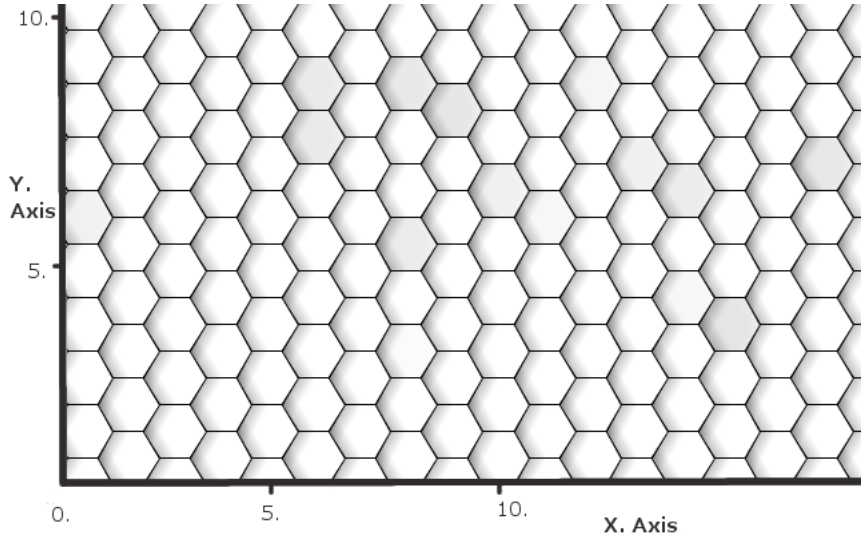

Figure 77: A typical hexagonal grid using Cartesian coordinates.

For many war game enthusiasts the hex grid is seen as the optimal grid type to use. In general three main grid types used in board/computer games, these are respectively triangle, square, and hexagonal grids. These shapes are most commonly used for their ability to tessellate, however this project has discounted both square and triangle grids under the following criteria.

1. Triangle grids make placing exact coordinates difficult, and since this project requires complex interaction between cells within a grid according to their coordinates this makes triangle grids less appealing.

2. In comparison hexagonal grids have more degrees of movement, and more degrees of relationship than square grids. This in particular is one of the reason many war games use the hexagonal grids.

3. A square as a structure is less "organic" as in it is not commonly found in nature, conversely a hexagonal grid can form more natural looking shapes. One way of making mitigating the unnatural shape of the square is to increase the resolution of the square grid, however this could cause significant slow down to the application. Considering this is using web browser technology, performance should be a priority.

4. Use of the Cartesian coordinate system will be vital to label each cell within the grid. The main issue with applying Cartesian coordinates to a hexagon grid will be the staggered offset of either the x or y coordinate depending on the orientation of the hex grid. In figure 77 the hex grid is orientated horizontally, whilst the x coordinate applies to hexagon cells in the straight line vertically, the y axis applies in a serpentine like pattern.

   Overall a hexagonal grid would be the best asset for the Coastaldefender project due to its ability to provide a greater complexity, for its compatibility with Cartesian grids and its ability to form natural seeming shapes.

8.2.4 Creating shapes within Google maps

Drawing polygons within Google maps is done via manual entry of coordinates into the path of a polyline or polygon object. To date no native function in the API exists to draw shapes. However the V3_e-shape.js library created by `http://econym.org.uk` provides the neccassary functions. This file was originally designed for Google Maps API v2.0 but was since updated by Lawerence Ross to port it into API 3.0.

```
1 google.maps.Polygon.RegularPoly = function(point,radius,
     vertexCount,rotation,strokeColour,strokeWeight,
     Strokepacity,fillColour,fillOpacity,opts,clickable) {
2         rotation = rotation||0;
3         var tilt = !(vertexCount&1);
4         return google.maps.Polygon.Shape(point,radius,radius,
             radius,radius,rotation,vertexCount,strokeColour,
             strokeWeight,Strokepacity,fillColour,fillOpacity,
             opts,clickable)
5 }
```

This constructor allows the creation of a single regular polgon within Google Maps. It uses the existing `google.maps.Polygon` method available in the API. The variables within the function mostly contain generic options such to change the shape of the polygon, however non standard variables exist.

Point: The variable determining the center of the regular polygon.

Radius: The radius of the regular polygon.

vertexCount: The number of vertices within the regular polgygon. I.E. a hexagon has six.

Rotation: Initially the rotation is set so the shape is vertically aligned.

```
1 function EOffsetBearing(point,dist,bearing) {
2     var latConv =  google.maps.geometry.spherical.
          computeDistanceBetween(point,new google.maps.LatLng(
          point.lat()+0.1,point.lng()))*10;
3     var lngConv =  google.maps.geometry.spherical.
          computeDistanceBetween(point,new google.maps.LatLng(
          point.lat(),point.lng()+0.1))*10;
4         var lat=dist * Math.cos(bearing * Math.PI/180)/
              latConv;
5         var lng=dist * Math.sin(bearing * Math.PI/180)/
              lngConv;
6         return new google.maps.LatLng(point.lat()+lat,point.
              lng()+lng)
7       }
```

The `EOffsetBearing` function allows one to position a new polygon in a position deviated from the initial variable of point. The additional two variables in this function `dist`, and `bearing` determine the new shapes' location by `dist` and the `bearing` respectively.

Within the function it calls methods from the API geometry library. These are necessary in API V3.0 as it takes into account that the surface of Google Maps is a geoid not a perfect sphere.

The `google.maps.geometry.spherical.computeDistanceBetween` method allows for the drawing of shapes taking into account the curvature of the Digtal Earth surface.

8.2.5 Creating the Hexagonal Grid

To place the hex grid on Google Maps functions had to be created using the V3_eshape.js file. These functions have to create, the rows for each hex, the overall grid and finally to create each hex object.

```
1 function newHex (g,i) {
2   this.row = g + 1;
3   this.column = i + 1;
4   var infoWindow= new google.maps.InfoWindow({
5     content: this.column+","+this.row,
6     });
7   marker = new google.maps.Marker({
```

```
8       position: EOffsetBearing(point,d*i,90),
9       map: map,
10      visible: false,
11      clickable: true,
12      title: this.column+","+this.row,
13    });
14    google.maps.event.addListener(hex, 'mouseover', function(ev
          ){
15      infoWindow.setPosition(ev.latLng);
16      infoWindow.open(map);
17    });
18    google.maps.event.addListener(hex, 'mouseout', function(ev)
          {
19      infoWindow.setPosition(ev.latLng);
20      infoWindow.close(map);
21    });
22  }
```

The `newHex` function creates a `newHex` object. At this point the hex object is not greatly complex, more a proof of concept. Variables `i` and `g` which are derived from the `buildRow` and `buildGrid` functions provides the Cartesian coordinates used to uniquely identify the hex.

In addition event listeners have been added to the hex objects. This is to show that each hex can be manipulated by the player to change the inherent state of the individual hex object.

```
1   function buildRow() {
2   var i=0;
3     while (i<10){
4         hex = new google.maps.Polygon.RegularPoly(
              EOffsetBearing(point,d*i,90),250,6,90,"#000000",1,1,
              "#00ffff",0.5,true);
5         hex.setMap(map);
6         newHex(g,i);
7         i++;
8         google.maps.event.addListener(hex,"mouseover",function
              () {
9           this.setOptions({fillColor: "#00FF00"});});
10        google.maps.event.addListener(hex,"mouseout",function()
              {
11          this.setOptions({fillColor: "#FF0000"}); });
12
13    }
14  }
```

The `buildRow` function creates a single row of `newHex` objects within the game map, as well as using the `google.maps.Polygon.RegularPoly` constructor to create hexagon shapes in a line on the map so that they are adjacent but not overlapping.

Each iteration of the while loop displaces the `newHex` object from the initial variable `point`, this is achieved by using the `EOffsetBearing` function in the constructors' attributes. With each iteration of the while loop the value of i is incremented by one thus displacing the position of the next `newHex` object by a multiples of the radius of the hex.

```
1  { function buildGrid(){
2  g=0;
3  var row=0;
4    while (g<10){
5      //detects if the row is starting row.
6      if (row == 0) {
7        buildRow();
8        g++;
9        row++;
10     }
11     //dectects if row is odd.
12     else if (row%2) {
13       point = (EOffsetBearing(point,d,210));
14       buildRow();
15       g++;
16       row++;
17     }
18     //detects if row is even.
19     else {
20       point = (EOffsetBearing(point,d,150));
21       buildRow();
22       g++;
23       row++;
24     }
25   }
26 }
```

The `buildGrid` displaces the position for each new row and calls the `buildRow` function. Each row is displaced from the previous. In addition each iteration of a new `buildRow` function being called increments g, this applies to the y axis label of Cartesian coordinates for each hex object.

The `buildGrid` function also has to determine the difference between odd and even rows. This is to identify how each row should be displaced to

aid tessellation. Displacement is performed by using the `EOffsetBearing` function.

8.2.6 Summary

Figure 78: The completed hexgrid, the info window shows the Cartesian coordinate of the highlighted hex.

Figure 78 shows the appearance of a drawn hex grid through this method. The hex grid gives proof of concept that a hex grid can be created on Google maps that can be manipulated in this manner. The next mission cycle for this project will be to build upon this by creating a conceptual model and prototype done via UML.

The prototype will be the software model broken down into its most basic elements, that are required to show that the game with its full mechanics can work.

8.3 2nd Cycle

8.3.1 Overview

With the completion of the basic hex grid a form and shape of the project as a whole can be considered. Since this project is being undertaken with the agile process in mind the initial game model is very much abstract and open to change. Indeed at the point of creation of the game prototype the game model will be revisited to consider more efficient design and potential obstacles that appear.

The aim of this specific mission is to create an abstract game model through the Unified Modelling Language (UML) and present this in both UML diagram and in a plain English description.

8.3.2 Objectives

The mission has a single primary objective.

1. **Create single player model:** To describe in a UML class diagram and in plain English how the game will work in detail.

This mission has two secondary objectives.

8.3.3 UML Class Diagram

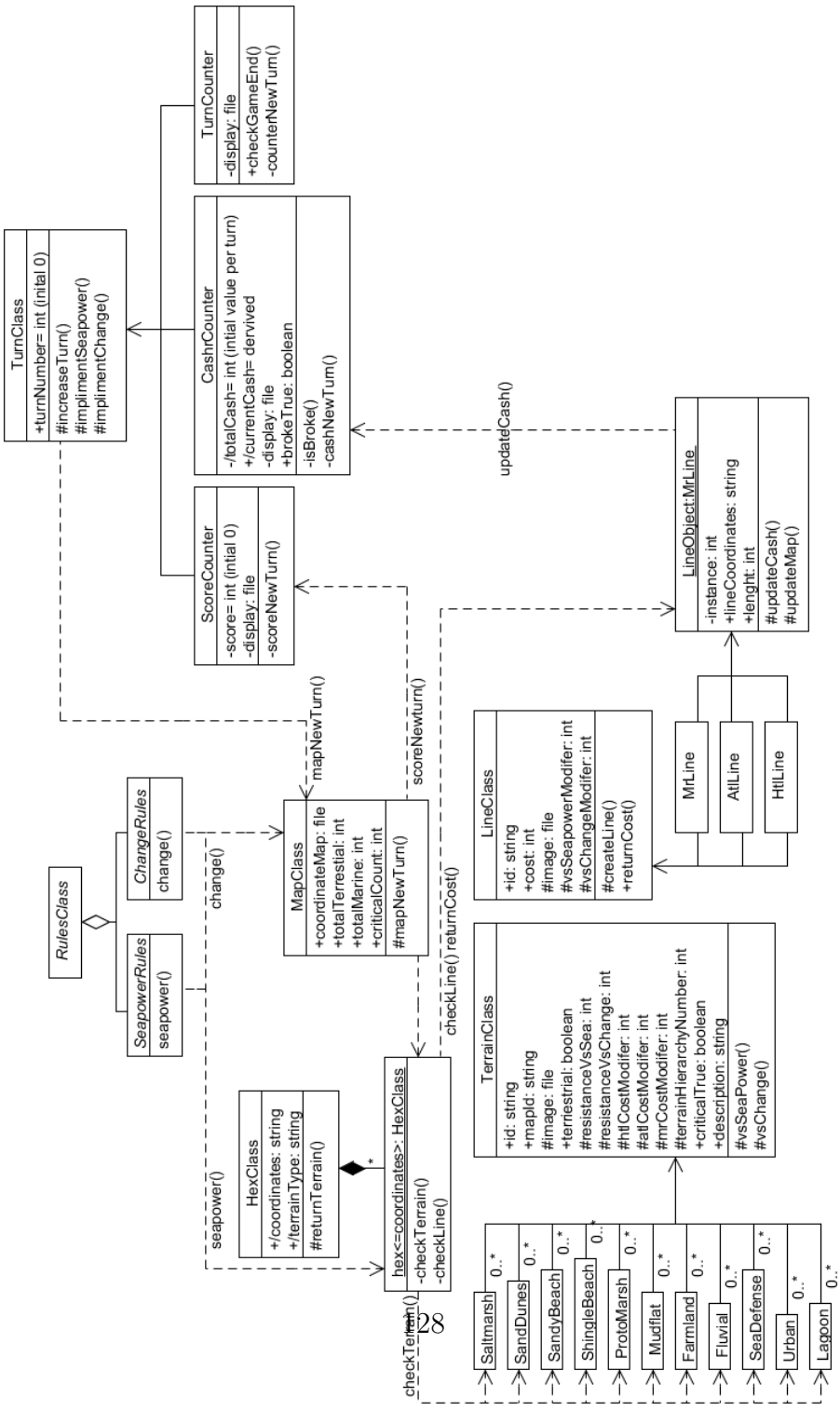

Figure 79: The Inital UML Class Diagram

`HexClass:` The `HexClass` governs the nature of each individual hex within the game, it stores information on its coordinates and terrain type (the terrain type governs its behaviour).

`TerrainClass:` The `TerrainClass` describes and governs the rules for how individual hex does behave. Different terrain types have different properties and the `TerrainClass` is an abstract super class that details all of the shared attributes that each individual terrain type has. In addition it stores the methods that describe how that terrain type responds to a specific rule.

`LineClass:` The `LineClass` describes the attributes of created lines during the game. Like the `TerrainClass` the `LineClass` is an abstract super class. The `LineClass` contains attributes which governs the cost of creating lines as well as their ability to modify a specific terrain type in response to specific rules.

`MapClass:` The `MapClass` governs the overall map information. It will keep record the number and location of different types of terrain within the map. The `MapClass` also records the number of critical locations in the game, thus being able to inform if the number of lost critical locations has exceeded the acceptable limit for game loss.

`TurnClass:` The `TurnClass` is a super class that governs three separate classes, the score counter, the cash counter, and the turn counter. The `TurnClass` again is a abstract class that keeps count of the current game turn whilst implementing methods that increase the turn, and applies rules to the game.

`RulesClass:` The `RulesClass` is a mostly empty abstract super class which has within its auspice both the change rule and the seapower rule, the two rules which test how the game will be affected each turn.

`TerrainClass` in Detail: The `TerrainClass` is an abstract super class. It has a zero to many association by generalisation, for its child subclasses. Each associated subclass is a type of terrain e.g. Marine, Beach, Salt marsh, Urban etc... None of the subclasses has additional attributes or methods outside of those inherited by the `TerrainClass`, these are.

- `id:` A public attribute. The id is a simple string used to identify that specific terrain type. For example the Salt marsh might have a `id` of `SaltM`.

- `mapId:` A public attribute. This is a simple string used to identify that specific terrain type on the game map. For example the Salt marsh terrain might have a `mapId` of `SM`.

- `image:` A public attribute. The image file attribute that corresponds to the classs terrain type.
- `terrestrial:` A public attribute. This is a Boolean value denoting if that terrain class is terrestrial or marine terrain.
- `resistanceVsSea:` A protected attribute. This is an integer value that denotes this terrain class resistance vs the seapower rule.
- `resistanceVsChange:` A protected attribute. This is an integer value that denotes this terrain class resistance vs the change rule.
- `htlCostModifier`: A protected attribute. This is an integer value that modifies the cost of a HTL line being placed upon that hex.
- `atlCostModifer:` A protected attribute. This is an integer value that modifies the cost of an ATL line being placed upon that hex.
- `mrCostModifer:` A protected attribute. This is an integer value that modifies the cost of a MR line being placed upon that hex.
- `terrainHierachyNumber:` A protected attribute. This is an integer that denotes the terrain class position within the terrain hierarchy. The terrain hierarchy values with the "most terrestrial" are the highest and "most marine" being the lowest. When the change or seapower rules are applied to a hex the terrain hierarchy number determines what terrain it can become. For example a salt marsh may become a beach or a dune but not farmland or marine.
- `criticalTrue:` A public attribute. This is a Boolean and determines where or not this is a critical location or not.
- `description:` A public attribute. This is a long string description of that terrain. For example a Salt marsh terrain would be described by its appearance, typical flora and fauna as well as its function as a coastal environment.

The `HexClass` has a dependency on the terrain type subclasses via the `checkTerrain` operation.
The `HexClass` in detail:

The HexClass contains two derived public attributes. The HexClass governs the spatial id of the hexes as well as applying a number of operations from other classes that it is dependent upon.

- Coordinates: A public derived attribute. This is a simple string containing x,y Cartesian coordinates, this will record the coordinates on the MapClass via the checkCoords operation.

- TerrainType: A public derived attribute. This attribute is derived from the terrain type subclass as well as the MapClass as it requires both the mapId and the coordinateMap to determine which terrain is at which coordinate, as well as the attributes associated with that subclass.

  The operations within the Hexclass determine a number of dependencies that this class has with others, these are.

- CheckTerrain: A private operation which checks the terrain type for this hex at the beginning of each turn. For this it needs to check against the MapClass CoordinateMap for the MapId of that hex.

- ReturnTerrain: A private operation. This then applies the correct terrain type subclass to that hex. This operation directly follows CheckTerrain.

- CheckLine: A private operation. At the end of each turn this is applied, it records which lines have been placed on this hex if applicable. This will then modify the attributes of the TerrainClass for that hex.

- CheckRules: A private operation. At the end of each turn checkRules is applied to see if this hex is required to take a seapower or change rule test.

The LineClass in Detail:

The LineClass is similar to the TerrainClass in its structure. The LineClass is an abstract superclass with three subclasses which derive all their traits from the superclass. Finally the lineObject is derived from one of these three subclasses, inheriting the attribute qualities of that particular subclass. The main LineClass attributes and operations are as follows.

- `Id:` This is a public attribute. The id is the type of line. In this case either HTL (hold the line), ATL (advance the line), or MR (managed realignment).
- `Cost:` This is a public attribute. The cost represents the total amount of cash/resource one hex of the line costs and appears as an integer. The cost varies depending on the type of line; the cost is then further modified by the type of terrain, which is determined by the `returnCost` operation.
- `Image:` This is a protected attribute. This is an image file associated with that line.
- `vsSeapowerModifier:` This is a protected attribute. The `vsSeapowermodifer` is an integer that modifies the terrain type subclass attribute of `resistanceVsSeapower`.
- `vsChangeModifer:` This is a protected attribute. The `vsChangermodifer` is an integer that modifies the terrain type subclass attribute of `resistanceVsChange`.
- `createLine:` This is a protected operation. It is this operation that creates a new line on the map and in the process creates a new `lineObject`.
- `returnCost:` This is a protected operation. As each hex is designated for a line the return cost operation is called. It derives the amount of cash/resources consumed by checking against the cost of the line and the cost modifier of that terrain type on which is was applied.
- `The LineClass: LineObject` is the object that inherits the traits from the `LineClass` and its subclass. It has some addition attributes and operations.
- `Instance:` this is a private attribute. This is represented by a simple integer, as each created line will be given an instance number in ascending order (starting with 0).
- `LineCoordinates:` This is a public attribute. The attribute is represented by a string of Cartesian coordinates one set for each hex it covers.
- `Length:` This is a public attribute. Represented by an integer this is the total number of hexes that the line covers. The length of the line directly affects the cost.

- `UpdateCash:` this is a protected operation. Once a line has been applied to each hex the cash is updated. This is a dependency for the `cashCounter` class.

- `ReturnCost:` This is a protected operation. Once a line designation has been completed this operation will be called to determine the total cost of the line. This is dependent on, the line subclass, the length and the terrain modifier.

The `MapClass` in detail:

The `MapClass` has no subclasses. It deals with the creation and subsequent changes in the game map at the beginning of and at the end of each turn. The `MapClass` also keeps details of which hex is which terrain type. Its attributes and operations are as below.

- `CoordainteMap:` This is a public attribute. The `CoordinateMap` is a file which tracks the current terrain types of each hex. This file could well appear as an excel spreadsheet as this file needs to record things as Cartesian coordinates in an x,y fashion. Where each cell of the spreadsheet represents a hex.

- `TotalTerrestial:` This is a public attribute. The total terrestrial is a simple integer that records how many of the existing hexes at the beginning of a turn are terrestrial. This will be used to determine score.

- `TotalMarine:` This is a public attribute. The total terrestrial is a simple integer that records how many of the existing hexes at the beginning of a turn are marine. This will be used to determine score.

- `TotalCritical:` This is a public attribute. The total terrestrial is a simple integer that records how many of the existing hexes at the beginning of a turn are Critical locations. This will be used to determine score.

- `MapNewTurn:` This is a protected operation. At the end of each turn the `MapNewTurn` operation is called. This operation will update the map hexes, by calling the `checkTerrain` operation from the `HexClass`. This will in turn provide the data needed to update the `CoordinateMap`.

The RulesClass in detail:

The RulesClass contains the two sub classes that govern the rules that are enacted at the end of each turn. These two rules are SeapowerRules and ChangeRules respectively, the MapClass and the HexClass have a dependency on these rules. Both subclass only has one operation this is Seapower and Change, these are the algorithms that enact the game rules within the hex grid.

The TurnClass in detail:

The TurnClass is a super class that has three child subclasses. The TurnClass keeps track of the current turn in the game and will be used to call operations in the RulesClass via the mapNewTurn operation. Its subclasses govern aspects of the game that requires the turn mechanics. The attributes and operations are as follows.

- turnNumber: This is a public attribute. The turnNumber appears as a simple intiger, its inital value is 0 (since the game has not yet started). The turnNumber increases by one at the end of each turn.
- increaseTurn: This is a public operation. The increaseTurn method increases the turn number at teh end of each turn.
- mapNewTurn This is a public operation. It is called at the increase of the turnNumber and updates the MapClass.

ScoreCounter:

The ScoreCounter is a subclass of the TurnClass. This appears as an object on the game map the score counter displays the current game score, this score is determined by the state of the MapClass.

- Score: Score is a private derived attribute; the attribute is derived by the ScoreNewTurn operation. The score attribute is a simple integer and the initial value is 0.
- Display: Display is a private attribute. Display is an image file with a counter inbuilt. The display will show the current score and this will be updated at the end of each turn when the ScoreNewTurn operation is called and completed.
- ScoreNewTurn: This is a private operation. The score new turn operation is called at the end of each turn and will update the current score by monitoring changes in the MapClass, CoordinateMap attribute.

CashCounter:

The Cashcounter is similar to the ScoreCounter in the manner in which it works. Using the CashNewTurn operation and the UpdateCash operation it keeps track of the amount of cash/resource the player has at any one time. It also through use of the isBroke operation checks to see if the player is out of cash/resources.

- TotalCash: TotalCash is a private derived attribute. This is an integer with the initial value of TotalCash is 0, and this is reset to zero at the beginning of each turn. TotalCash is also updated each time a new line is created via the UpdateCash operation.
- Display: The Display is a private attribute. The display is an image file along with a counter that will change according to the total cash value.
- BrokeTrue: This is a private attribute. BrokeTrue is a Boolean; it is initially set to false. As long as the Cashcounter does not equal 0 or less BrokeTrue remains false. Once this attribute is true it calls the IsBroke operation.
- IsBroke: This is a protected attribute. If the player attempts to expend cash whilst IsBroke true equals true the IsBroke operation is called. This prevents the player from expending further cash.
- CashNewTurn: this is a private operation. The cash new turn operation resets the TotalCash to zero.

TurnCounter:

The TurnCounter is an object similar to the cash and score counter. It displays the current turn on the map. It also records if the end of the game limit has been reached.

8.4 3rd Cycle

8.4.1 Overview

The turn button initiates the rules which change the game state after the player has made their game play decisions for that turn. Eventually the turn button will essential call a number of AJAX functions which will access the

back end game logic engine. However since this is currently at the prototyping state the purpose of the turn button will be to cause change in the game view on Google Maps.

The aim of this specific mission is to create the "turn button" and associated functions that cause a change to the game state. This will include changes to hex objects as well as the view on Google Maps where the nature of hexes will change according to the rules in the game logic engine.

8.4.2 Objectives

This mission has three primary objectives.

- **Refresh:** To refresh the map so that it can be altered by the next invocation of the turn button. It should also work so that when lines are added in the future mission Line Creation and Designation these will also be removed from the game state allowing the player to make new game play decisions.
- **Turn button object:** To create the button as an object on the webpage, clicking this button will invoke the functions associated with turn progression.
- **Seapower basic:** To create a prototype of the Seapower function where a simple rule will be applied to existing terrestrial hexes causing them to change into marine hexes. This will demonstrate how Seapower will work in the finished game.

This mission has three secondary objectives. Due to the prototype nature of this stage of the projects development it is unlikely that these more advanced and auxiliary aspects of the game will be created at this stage of development.

- **Move the turn on:** Simply to change the counter on the number of turns by one for each call of the turn button.
- **Seapower advanced:** A more advanced version of the Seapower rule and associate functions, where a number of more complex changes to the game state will occur.

- **Land change**: Similar to Seapower Land change is an advanced function of the game where the environment of the terrestrial hexes takes place according to progression or regression in the terrain Hierarchy.

8.4.3 Amending the Hex Objects

For the turn button to apply changes to the hex grid and game state the `NexHex` objects in the hex grid require a number of new attributes. These additional attributes will govern the terrain type of the hex as well as its appearance on the map (its colour). The `NewHex` objects will contain the new attribute of terrain, referring back to the UML class diagram description terrain can be considered a distillation of the terrain class, a class which describes the behaviour of a hex. In the case of the prototype terrain determines if the hex is terrestrial or marine. The terrain attribute is determined by the elements a 2D array called `mapList`; in this case the value of terrain is value of `mapList` which shares the `NewHex` objects Cartesian coordinate. Therefore the `NewHex` constructor appears as this.

```
1 function NewHex (g,i) {
2   this.row = g;
3   this.column = i;
4   this.id = null;
5   this.terrain = mapList[g][i];
6   this.hex = new google.maps.Polygon.RegularPoly(
        EOffsetBearing(point,d*i,90),250,6,90,"#000000",1,1,"#00
        ffff",0.5,true);
7   this.hex.setMap(map);
8 }
```

The 2D array `mapList` is a global variable.

```
1  var mapList = [
2  ["m","m","m","m","m","m","m","m","m","m"],
3  ["m","m","m","m","t","t","t","t","t","t"],
4  ["t","m","m","m","t","t","t","t","t","t"],
5  ["t","t","m","m","m","t","t","t","t","t"],
6  ["t","t","t","t","m","t","t","t","t","t"],
7  ["t","t","t","t","m","t","t","t","t","t"],
8  ["t","t","t","t","t","t","t","t","t","t"],
9  ["t","t","t","t","t","t","t","t","t","t"],
10 ["t","t","t","t","t","t","t","t","t","t"],
11 ["t","t","t","t","t","t","t","t","t","t"]
12 ];
```

In addition to the terrain attribute, the `updateOptions` method is also applied to each hex. The `updateOptions` method is applied after the creation of the hex grid and is also called via the turn button at the end of each turn. The method changes the current colour of the hex according to its current terrain attribute. The `updateOptions` method is applied via JavaScript prototype, where a single method is attached to a constructor. This method also opens an info window on a `mouseover` event to show the current terrain attribute of the `NewHex` object at that specific point in the hex grid.

```
1  NewHex.prototype.updateOptions = function(){
2  var fillColour;
3    if (this.terrain == "m")
4      fillColour = "#0000ff";
5    else if (this.terrain == "t")
6      fillColour = "#00ff00";
7    this.hex.setOptions({fillColor:fillColour});
8
9    var infoWindow = new google.maps.InfoWindow({
10     content: this.terrain,
11     });
12     google.maps.event.addListener(this.hex, 'mouseover',
           function(ev){
13     infoWindow.setPosition(ev.latLng);
14     infoWindow.open(map);
15   });
16   google.maps.event.addListener(this.hex, 'mouseout',
         function(ev){
17     infoWindow.setPosition(ev.latLng);
18     infoWindow.close(map);
19   });
20 }
```

8.4.4 Adding the Turn button

The button will be added to the main HTML front page as a simple Javascript button. Each time the button is clicked it will call the `newTurnMap` function.

```
1  <button type="button" onclick="newTurnMap()">Advance Turn!</
       button>
```

8.4.5 The process of change

The process of changing the game state requires a number of steps.

1. Check the terrain attribute for each member of the hex grid.

2. If the terrain attribute of a given element of the hex grid is terrestrial then it is checked to see if that hex is adjacent to a marine hex.

3. If that hex is adjacent to a marine hex it is then converted into a marine hex by amending the terrain attribute of that member of the hex grid.

8.4.6 Checking Attributes

The `newTurnMap` function is used to check through the entire hex grid via 2D for loop, which checks each member of the hex grid in turn to see if the terrain attribute of that object is terrestrial (marked as `t`).

```
function newTurnMap(){
  for( x = 1 ; x<9;x++){
    for(y = 1 ; y<9;y++){
      if (myGameGrid.gameGrid[x][y].terrain == "t"){
          if (isMarine(x,y)){
          myGameGrid.gameGrid[x][y].modified = true;
        }
      }
    }
  }
  changeTerrain();
}
```

The `newTurnMap` function is the highest level function in the process of change on the map. If the member of hex grid is found to have its terrain attribute equal to `t` it will then call the `isMarine` function. The `isMarine` function returns a boolean, if it returns true the hex grid member modified attribute is mutated to true.

The `isMarine` function tests every adjacent hex to the hex grid member flagged as `t` to see if the terrain attribute of that hex is marine (flagged as `m`). It does this by looping through a 2D array of two for loops so it incidentally also checks the flagged hex itself, however because the initial hex is always `t` it does not erroneously return `isM`, True.

```
function isMarine(x,y){
    var isM = false;
  for (var i = -1; i < 2; i++){
      for (var j = -1;  j < 2; j++){
          if (myGameGrid.gameGrid[x+i][y+j].terrain == "m"){
```

```
            isM = true;
        }
      }
    }
    return isM;
  }
```

8.4.7 Changing g Attributes

Once the appropriate `NexHex` objects have been identified to be changed via the `updateOptions` method a new function must be called to change all the `NewHex` objects with terrain attribute of `t` to `m`. However applying `updateOptions` without some how flagging the appropriate `NexHex` objects to be changed would cause an instantaneous change of all hexes in the grid to `m`. Therefore a new attribute to `NewHex` constructor called modified, it has a default value of false and if `isMarine` returns True, the `newTurnMap` function mutates this attribute to True.

```
function NewHex (g,i) {
  this.row = g;
  this.column = i;
  this.id = null;
  this.terrain = mapList[g][i];
  this.modified = false;
  this.hex = new google.maps.Polygon.RegularPoly(
     EOffsetBearing(point,d*i,90),250,6,90,"#000000",1,1,"#00
     ffff",0.5,true);
  this.hex.setMap(map);
}
```

Once the `newTurnMap` function has mutated the modified attribute for all of the appropriate `NewHex` objects it calls the `changeTerrain` function. The `changeTerrain` function again uses two for loops to cycle through the 2D hex grid array. For each member of the array it checks if the modified attribute is equal to false, is this condition is true it mutates the terrain attribute from `t` to `m`. It mutates the modified attribute to false so at the next call of the `newTurnMap` function it will be starting afresh and can progressively apply the seapower rules to the game. Lastly the `changeTerrain` function calls the `updateOptions` method which changes the appearance of all of the flagged `NewHex` objects.

```
function changeTerrain () {
```

```
2     for (var i = 0; i<10; i++) {
3       for (var j = 0; j<10; j++) {
4         if (myGameGrid.gameGrid[i][j].modified == true) {
5           myGameGrid.gameGrid[i][j].terrain = "m";
6           myGameGrid.gameGrid[i][j].modified = false;
7           myGameGrid.gameGrid[i][j].updateOptions();
8         }
9       }
10    }
11  }
```

This completes the process of changing the map and satisfied the requirement of allowing the objects to be in a refreshed state so that the next calling of the turn button the process can start again allowing for a dynamically changing game map and grid.

8.4.8 Summary

The methods employed in the process of changing the game state are similar to the creation of the grid in the first place. That is to say it uses nested loops to navigate through the 2D array that is the hex grid. This part of the project also takes advantage of object orientated programming; it uses the mutability of persistent objects to store and change the game state. At this stage in the project it is possible to consider that the process of checking and changing the map can be largely standardised. The process will work as thus.

1. To navigate the 2D array hex grid via nested loops.
2. Check the value of an attribute for the `NewHex` object at that given Cartesian coordinate.
3. Mutate the appropriate attribute of the `NewHex` object to denote it.
4. Navigate the hex grid again via nested loops.
5. Mutate the appropriate attributes of the flagged `NewHex` objects.
6. Mutate the attribute of the `NewHex` object that denotes if it has been flagged for change to its default value.

8.5 4th Cycle

8.5.1 Overview

Line creation and designations will be the primary game mechanic; ultimately the player using a limited number of resources will designate hexes according to their policy decisions. This designation will change the behaviour of the hex object according to policy type of the line chosen. At the prototyping stage of the project, the line will simply interdict any change of the terrain attribute for the designated hex.

8.5.2 Objectives

This mission has three primary objectives.

1. **Blank line overlay:** To create an overlay of a line on the map when a single `NewHex` object in the hex grid is selected via a mouse click.
2. **Line creation:** To create a new instance of a line object which is created and extended upon by the player as they designate additional hexes. The instance of the line object must be deleted at the end of the turn to avoid clashes in future turns.
3. **Line designation:** To designate the existence of a new line on the hex grid via a mouse click. Each additional line designation is only able to be made to adjacent hexes.

This mission has one secondary objective, currently this objective will not be considered as it is not clear how the lines will work and be created at this point in time.

1. **Line object:** To create a database of line objects so that this can be recreated on a new game map.

8.5.3 Amending the Hex Objects

In the previous mission the modified attribute was added to the `NewHex` constructor, its purpose is to flag a specific member of the grid allowing for its terrain attribute to be changed via the `changeTerrain` function. Line creation will use a similar process. A new attribute of `hasLine` will be added to determine if that hex grid member has a line designated to it. The `hasLine` attribute is a Boolean, and is by default set to false.

```
function NewHex (g,i) { ...
this.modified = false;
...
```

In addition a new global variable is created called `lineExists`. This monitors the existence of any lines on the map. It is by default set to false.

```
var lineExists = false;
```

As stated in the primary objectives for this mission. Line creation is called via a mouse click on a given member of the hex grid. This is achieved via a listener tied to a right click of the mouse button on a specific `NewHex` object.

```
NewHex.prototype.lineStuff = function() {
  var r = this.row;
  var c = this.column;
  google.maps.event.addListener(this.hex, 'rightclick',
     function(event) {
    buildLine(r,c);
  });
}
```

The listener is created via Javascript Prototype function in the same manner as the `updateOptions` method which is part of the `NewHex` constructor discussed in the Turn button and Change mission. The listener creates two variables based off the Cartesian coordinates and passes them as arguments to the `buildLine` function. The `buildLine` function is called by the activation of the listener, which is linked to a right click of a mouse on the `NewHex` object.

8.5.4 Building the Line

The line will be created as a `Polyline` object, which is a standard Google Maps API object. The `Polyline` constructor is almost identical to the Polygon constructor that is used by the `regularPoly` function in `V3_eshapes.js` to draw the hex grid as described in the first cycle

The `buildLine` function has the coordinates of the hex grid member clicked passed to it as arguments via the `lineStuff` listener, the first action of the function is to create the `newPoint` variable that is a `LatLng` centred on the clicked hex.

The function checks against the `lineExists` global variable. If no line currently exists on the map and `lineExists` is set to false a new `Polyline` object is created, this line is named `theLine`. In addition the `buildLine` function changes the `hasLine` attribute of the `NewHex` object to true.

The path for the `Polyline` uses the `getPath` method to retrieve an empty path attribute and appends (via the push method) the `LatLng` attached to the `newPoint` variable to the path array. Finally it sets the `lineExists` global variable to true, so that further calls of the `buildLine` function test false for the first if conditional.

```
1  function buildLine(r,c) {
2  var newPoint = myGameGrid.gameGrid[r][c].position;
3
4    if (lineExists == false) {
5      var polyOptions = new google.maps.Polyline({
6        strokeColor: '#000000',
7        strokeOpacity: 1.0,
8        strokeWeight: 3,
9      });
10
11     theLine = new google.maps.Polyline(polyOptions);
12     theLine.setMap(map);
13     myGameGrid.gameGrid[r][c].hasLine = true;
14     var linePoints = theLine.getPath();
15     linePoints.push(newPoint);
16
17     var marker = new google.maps.Marker({
18       position: newPoint,
19       title: '#' + r + "," + c,
20       map: map
21     });
22
23     return lineExists = true;
24
25   }
26   if (checkAjacent(r,c)) {
27     var linePoints = theLine.getPath();
28     linePoints.push(newPoint);
29     myGameGrid.gameGrid[r][c].hasLine = true;
30
31     var marker = new google.maps.Marker({
32       position: newPoint,
33       title: '#' + r + "," + c,
34       map: map
```

```
35      });
36    }
37    else if (checkAjacent(r,c) != true) {
38      alert( "Hex needs to be ajacent");
39    }
40  }
```

Additional members of the hex grid that are clicked will be checked against the `checkAjacent` function. If the condition returns true the `getPath` function is used again to append a new coordinate onto the end of the `Path` array attribute, and the `hasLine` attribute is changed to false. If`checkAjacent` returns false an alert box is raised informing the player that their selection needs to be an adjacent hex.

```
1  function checkAjacent(r,c) {
2  var isAjacent = false;
3    for (var i = -1; i < 2; i++){
4        for (var j = -1;  j< 2; j++){
5            if (myGameGrid.gameGrid[r+i][c+j].hasLine == true){
6          isAjacent = true;
7        }
8      }
9    }
10 return isAjacent;
11 }
```

The `checkAjacent` function works in a similar manner to the `isMarine` function discussed in Turn button and Change mission. It uses nested loops to check the `hasLine` attribute for each adjacent hex, and will return true is one of the `hasLine` attributes is set to true.

8.5.5 Refreshing the Map

Removing the line from the map is a relatively simple operation; at the end of the `newTurnMap` function after the `changeTerrrain` function has been called `theLine` is assigned to null effectively deleting it. In addition it returns the global variable of `LineExists` as false.

```
1  function newTurnMap(){
2    for( x = 1 ; x<9;x++){
3      for(y = 1 ; y<9;y++){
4        if (myGameGrid.gameGrid[x][y].terrain == "t"){
5            if (isMarine(x,y)){
6            myGameGrid.gameGrid[x][y].modified = true;
```

```
7           }
8         }
9       }
10    }
11
12    changeTerrain();
13    theLine.setMap(null);
14    return lineExists = false;
15  }
```

8.5.6 Summary

Since the creation and designation of new points in a line will be the main game mechanic care has to be taken over the emergence of bugs and unintended results. For example if clicking on a hex which already has a line in it if adjacent to a hex with a line on it raises no errors, whilst at this stage of the project this is not an issue in the future when line creation mutates integer values of `NewHex` objects erroneous results may well arise. More conditionals should be considered.

Also it was apparent that there may be room for creating a catch all dynamic function that allows for checking values of adjacent hexes. Currently `isMarine` and `checkAjacent` are almost identical in their purpose, in the future it should be possible to create a single function that using the appropriate arguments will be able to perform both roles.

8.6 5th Cycle

8.6.1 Overview

The Costaldefender project needs the support of an appropriate web based framework. At its core Costaldefender can be consider a "Geo-Web" based product, in that it merges aspects of the traditional internet with the emerging titan of location based data. The Django web framework has been chosen for use in this project. Django is a relatively young framework but benefits from tremendous popularity amongst developers and a strong community. In addition it should be noted that Django is an in built GIS libary, which could be taken advantage of during this project.

8.6.2 Objectives

This mission has three primary objectives, with no secondary objectives.

1. **Learn Django:** To understand the Django Framework in all its details. Whilst it appears as the perfect solution for an appropriate web frame work for the project, it may become apparent that incompatibilities could emerge.

2. **Create a development site:** Django has an inbuilt development server, the initial website can be constructed via this before it is converted into a production server.

3. **Create a full working site:** To find a web host for the project and move from the development environment to the full production environment.

8.6.3 Website Structure

Starting a new Django project creates a folder and file structure as that seen below.

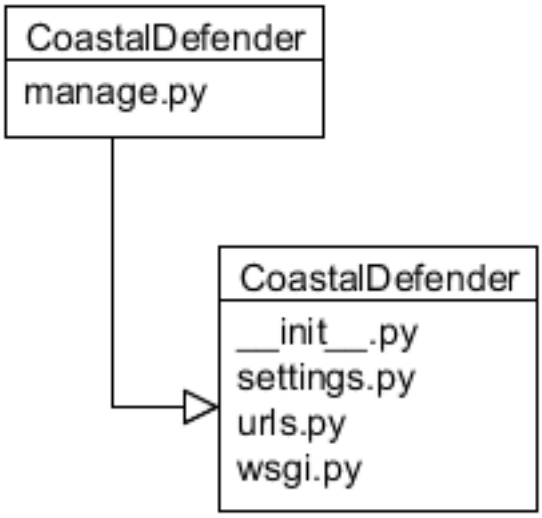

Figure 80: Django site structure

Any additional apps will be added to this main Coastaldefender folder with a structure similar to the Coastaldefender subdirectory. Specifically in the case of this project the game will be created in a Map app subdirectory. In addition folders for the web page templates needs to be created as well, these will be described in detail in subsequent parts of this mission. Therefore the overall beginning file structure will appear as thus.

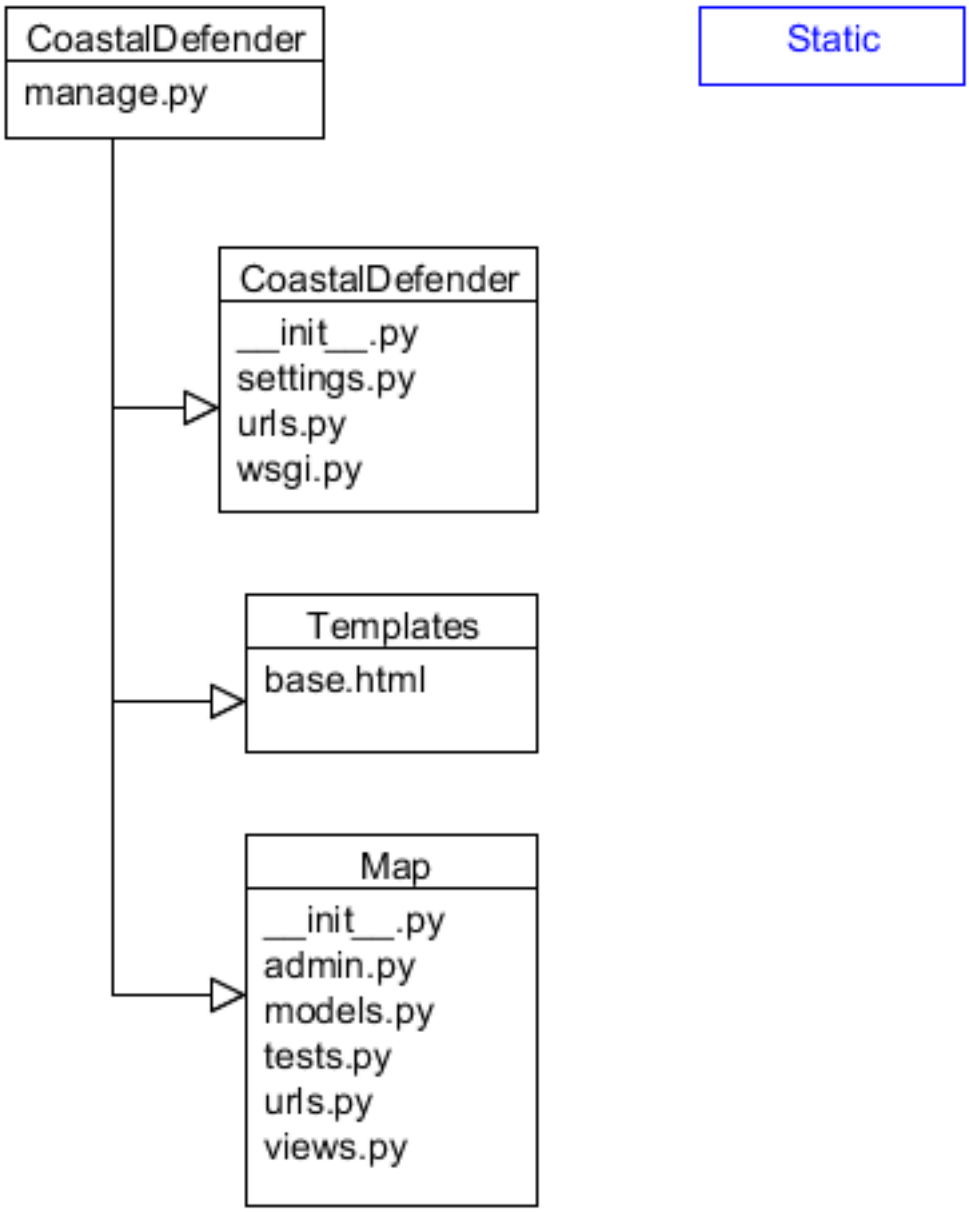

Figure 81: Django app structure.

`__init__.py`

An empty file that tells Python this directory belongs to a python package. Largely irrelevant to this project apart from its necessity of being to make Django work correctly.

`Wsgi.py`

Another largely irrelevant file for this project. `Wsgi.py` allows for an entry point for WSGI-compatible webservers.

`Settings.py:`

`Settings.py` as the name suggest contains all of the site specific settings and configurations for the project. It defines the root path for the static folder, as well as the apps associated with this project.

```
1  DEBUG = True
2  TEMPLATE_DEBUG = DEBUG
3
4  ADMINS = (
5      ('benjirama', 'b.b.a.mottram@googlemail.com'),
6  )
7
8  MANAGERS = ADMINS
9
10 DATABASES = {
11     'default': {
12         'ENGINE': 'django.db.backends.postgresql_psycopg2', #
               Add 'postgresql_psycopg2', 'mysql', 'sqlite3' or
              'oracle'.
13         'NAME': 'benjirama_coast',                      # Or
              path to database file if using sqlite3.
14         'USER': 'benjirama_coast',                      # Not
               used with sqlite3.
15         'PASSWORD': 'ea1877a6',                  # Not used
              with sqlite3.
16         'HOST': '',                      # Set to empty
              string for localhost. Not used with sqlite3.
17         'PORT': '',                      # Set to empty
              string for default. Not used with sqlite3.
18     }
19 }
20
21 # system time zone.
22 TIME_ZONE = 'Europe/London'
23
24 # Language code for this installation.
25 LANGUAGE_CODE = 'en-us'
```

```
26
27 SITE_ID = 1
28
29 # to load the internationalization machinery.
30 USE_I18N = True
31
32 # calendars according to the current locale.
33 USE_L10N = True
34
35 USE_TZ = True
36
37 # Absolute filesystem path to the directory that will hold
      user-uploaded files.
38 MEDIA_ROOT = ''
39
40 # URL that handles the media served from MEDIA_ROOT.
41 MEDIA_URL = ''
42
43 # Absolute path to the directory static files should be
      collected to.
44 STATIC_ROOT = '/home/benjirama/webapps/static'
45
46 # URL prefix for static files.
47 STATIC_URL = 'http://benjirama.webfactional.com/static/'
48
49 # Additional locations of static files
50 STATICFILES_DIRS = ()
51
52 # List of finder classes that know how to find static files
      in
53 # various locations.
54 STATICFILES_FINDERS = (
55     'django.contrib.staticfiles.finders.FileSystemFinder',
56     'django.contrib.staticfiles.finders.AppDirectoriesFinder'
          ,
57 )
58
59 # Make this unique, and don't share it with anybody.
60 SECRET_KEY =
61
62 # List of callables that know how to import templates from
      various sources.
63 TEMPLATE_LOADERS = (
64     'django.template.loaders.filesystem.Loader',
65     'django.template.loaders.app_directories.Loader',
```

```
66      'django.template.loaders.eggs.Loader',
67  )
68
69  MIDDLEWARE_CLASSES = (
70      'django.middleware.common.CommonMiddleware',
71      'django.contrib.sessions.middleware.SessionMiddleware',
72      'django.middleware.csrf.CsrfViewMiddleware',
73      'django.contrib.auth.middleware.AuthenticationMiddleware'
            ,
74      'django.contrib.messages.middleware.MessageMiddleware',
75  )
76
77  ROOT_URLCONF = 'myproject.urls'
78
79  # Python dotted path to the WSGI application used by Django's
        runserver.
80  WSGI_APPLICATION = 'myproject.wsgi.application'
81
82  TEMPLATE_DIRS = (
83      '/home/benjirama/webapps/map/myproject/templates'
84  )
85
86  #FORCE_SCRIPT_NAME reroutes the app to the correct directory
       on a webfaction website
87  FORCE_SCRIPT_NAME = '/map'
88
89  INSTALLED_APPS = (
90      'django.contrib.auth',
91      'django.contrib.contenttypes',
92      'django.contrib.sessions',
93      'django.contrib.sites',
94      'django.contrib.messages',
95      'django.contrib.staticfiles',
96      'django.contrib.admin',
97      'django.contrib.admindocs',
98      'map',
99  )
100
101 # A sample logging configuration. The only tangible logging
102 # performed by this configuration is to send an email to
103 # the site admins on every HTTP 500 error when DEBUG=False.
104 # See http://docs.djangoproject.com/en/dev/topics/logging for
105 # more details on how to customize your logging configuration
        .
106 LOGGING = {
```

```
107     'version': 1,
108     'disable_existing_loggers': False,
109     'filters': {
110         'require_debug_false': {
111             '()': 'django.utils.log.RequireDebugFalse'
112         }
113     },
114     'handlers': {
115         'mail_admins': {
116             'level': 'ERROR',
117             'filters': ['require_debug_false'],
118             'class': 'django.utils.log.AdminEmailHandler'
119         }
120     },
121     'loggers': {
122         'django.request': {
123             'handlers': ['mail_admins'],
124             'level': 'ERROR',
125             'propagate': True,
126         },
127     }
128 }
129
130 TEMPLATE_CONTEXT_PROCESSORS = ("django.contrib.auth.
        context_processors.auth",
131                                "django.core.
                                    context_processors.debug",
132                                "django.core.
                                    context_processors.i18n",
133                                "django.core.
                                    context_processors.media",
134                                "django.core.
                                    context_processors.static",
135                                "django.core.
                                    context_processors.request"
                                    ,
136                                "django.contrib.messages.
                                    context_processors.messages
                                    ")
```

`Settings.py` is a large document, and the majority of the does not require specific augmentation for a website, the parts that are will be described in greater detail.

`STATIC_ROOT` and `STATIC_URL`: Django gathers all static files into a a single folder from its various apps. It then serves the static files from this folder.

Static files are considered to be all non dynamic python script excluding HTML documents (these are served from the templates folder), this includes all JavaScript files, CSS documents, txt files and image files. `STATIC_ROOT` determines the path from which static files are served from, while `STATIC_URL` states the prefix URL when using template tags to serve static files.
`FORCE_SCRIPT_NAME`: This is a necessary addition whilst the Costal Defender is hosted with Webfaction `https://www.webfaction.com/` due to the manner in which Webfaction sets up its server side files. It allows the `/map` prefix to be set as the home directory when required.
`ROOT_URLCONF:` This sets the file where Django first checks when attempting to serve a specific URL.
`TEMPLATE_DIRS:` File path for the template folder in which HTML templates are stored.
`INSTALLED_APPS:` In addition to the default Django apps required for the web framework to function correctly any user created apps must be added to the python list. Without doing so any data or Urls associated with the app will not be recognised. In addition if `django.contrib.admin` and `django.contrib.admindocs` are applied to the list the Django admin site is enabled.
`TEMPLATE_CONTEXT_PROCESSORS:` A tuple of callables which are used in the `RequestContext` during http requests. These are necessary when using template tags.
`urls.py`

Acts as the URL dispatcher for the site. `urls.py` dispatches URL patterns to specific python callback functions. HTTP requests are handled by the following format.

1. An HTTP request is sent to the python file designated by `ROOT_URLCONF`.
2. Django searches for the `urlpatterns` variable in `urls.py`.
3. Django searches through each member of the `urlpatterns` list till it matches the requested URL.
4. When the Regexes match Django imports the view from the `views.py` and returns it as an argument in an HTTP Resquest.

```
1 from django.conf.urls import patterns, include, url
```

```
2  from django.contrib.staticfiles.urls import
       staticfiles_urlpatterns
3  from django.conf import settings
4  from django.conf.urls.defaults import *
5
6  from django.contrib import admin
7  admin.autodiscover()
8
9  urlpatterns = patterns('',
10
11     url(r'^admin/doc/', include('django.contrib.admindocs.
           urls')),
12     url(r'^admin/', include(admin.site.urls)),
13     url(r'^map/', include('map.urls')),
14 )
15
16 urlpatterns += staticfiles_urlpatterns()
17
18 if settings.DEBUG:
19     urlpatterns += patterns (
20     (r'^(?P<path>.*)$', 'django.contrib.staticfiles.views.
           serve'),
21     )
```

The Regexes in the `urlpatterns` tuple are created using regular expressions. In the case of the map app all associated urls are sent to a local `urls.py` within the map app folder. This allows for greater modularisation of URL handling and avoids the pitfalls of creating a single vast file that is hard to manage.

8.6.4 Map Subdirectory

The map subdirectory contains all the files for the Django app `map` any static files associated with the app are initially stored here though they are collected into the `ROOT_STATIC` designated folder to be served via a HTTP request.

Django apps are a collection of models and views that are returned as arguments in HTTP Requests/Responses. They are a powerful way to create a pool of resources that can be called by HTML template documents, in any number and any order.

`__init__.py`

An empty file that tells Python this directory belongs to a python package.

`Admin.py`

Handles the importing of views from the map app, allowing them to become accessible in the Django admin site.

`Models.py`

Models that will be used by Django views to create objects within the HTML document templates. Models are constructed as a python object and its attributes govern the editable values that appear in the Django admin page.

```
1 from django.db import models
2
3 class mapFrame (models.Model):
4     frame = models.CharField(max_length=20)
5
6     def __unicode__(self):
7         return self.frame
```

The sole model that exists currently is the `mapFrame` model. its design is deliberately minimalist since the `mapFrame` is the object into which the Google Maps window will be placed and serves no other purpose. However it does show the construction of a Django model in its most simple and understandable form. The `mapFrame` model is created as a python class, its attribute `frame` provides an editable`Charfield`, allowing the model to be displayed on a HTML template as a short line of text. However for the purposes of this project the`mapFrame` will be used as an empty object that will literally frame the Google Map map. The `__unicode__()` method is used to help created representations of the `mapFrame` object within the Django admin, it is considered best practice to use this with all models.

`tests.py`

A file used to create helpful tests to run when using Django.

`Views.py`

The `views.py` file contains views will will return how models appear within the context of an HTML template document.

```
1 #Django Imports
2 from django.shortcuts import render_to_response
3 #from django.template import request context
4 from django.template import Context, loader
5 #Site Imports
6 from django.http import HttpResponse
7 from map.models import mapFrame
8
```

```
def buildFrame(request):
    mapView = mapFrame
    t = loader.get_template('mappage.html')
    c = Context({
        'mapView': mapView,
    })
    return HttpResponse(t.render(c))
```

`Urls.py`

This url dispatcher works in exactly the same way as the main `urls.py` found in the main project folder. The initial `urls.py` redirects request to this file to search for appropriate regexes. I this case it imports views directly from the map app `views.py`.

```
from django.conf.urls.defaults import *

import views

urlpatterns = patterns('',
    url(r'^$', 'map.views.buildFrame'),
)
```

8.6.5 Templates Subdirectory

All HTML templates are stored in the templates folder, in Django templates follow the DRY (dont repeat yourself) principle by having a base template and further templates extending from it. The templates use Django templating language to achieve this. The templates folder currently contains two files base.html and `mappage.html` which extends from it.

Using Template tags

Django templating language is not particularly complex, it works by using tags to create a block into which content can be put into via child html documents. The below example would be placed on the parent HTML.

```
{% block exampleBlock %}
{% endblock %}
It will then be refered to in the child HTML.
{% block exampleBlock %}
<p> Helllo World </p>
{% endblock %}
```

This will emplace the “Hello World!” when the URL linking to the child HTML is called. The page will then be displayed using the structure of the parent HTML but using the content as defined by the child HTML.

`Base.html`

```
<html>
  <head>
    {% load staticfiles %}
    <link rel="stylesheet" type="text/css" href="{% static "
       blogstyle.css" %}">
    <script type="text/javascript" src="http://code.jquery.
       com/jquery-1.8.0.min.js"></script>
    {% block extrahead%}
    {% endblock %}
  </head>

  <body>
    <div id='main'>
        {% block content %}
        {% endblock %}
    </div>
  </body>
  {% block controls %}
  {% endblock %}
</html>
```

This `base.hmtl` document immediately uses the % load staticfiles % tag, this uses the URL as designated by `STATIC_URL` in `settings.py` to give the template access to all the associated static files in the URL.

It creates three blocks, extrahead, content and controls.

`Mappage.html`

```
{% extends 'base.html' %}
{% block extrahead %}
{% load staticfiles %}
    <meta http-equiv="content-type" content="text/html;
       charset=UTF-8"/>
    <title>Hex Grid</title>
  <script type="text/javascript" src="http://maps.google.com/
     maps/api/js?sensor=false&amp;libraries=geometry"></
     script>
  <script src="{% static "v3_eshapes.js" %}" type="text/
     javascript"></script>
  <script src="{% static "mapOperations.js" %}" type="text/
     javascript"></script>
```

```
{% endblock %}


{% block content %}

    <div id="map" style="width: 1024px; height: 800px;"></div>
  <div id="w3valid">
    <a href="http://validator.w3.org/check?uri=referer" ><!--
        target="validationresults" --><img
      src="http://www.w3.org/Icons/valid-xhtml10"
      alt="Valid XHTML 1.0 Transitional" height="31" width="
        88" /></a>
  </div>

    <noscript><p><b>JavaScript must be enabled in order for
       you to use Google Maps.</b>
      However, it seems JavaScript is either disabled or not
         supported by your browser.
      To view Google Maps, enable JavaScript by changing your
          browser options, and then
      try again.</p>
    </noscript>


    <script type="text/javascript">
Script for creating and setting up the inital map and hex
   grid for the game.
    </script>
{% endblock %}

{% block controls %}
  <button type="button" onclick="newTurnMap()">Advance Turn!<
     /button>
{% endblock %}
```

This child HTML provides content for the three blocks expressed in `base.html`. The extrahead block uses another % load static % tag to load additional static files to `mappage.html`. The content block inputs the JavaScript which controls the creation and initialisation of the map and hex grid. The controls block inputs the html and JavaScript button that calls the `newMapturn` function.

8.6.6 Summary

With the completed backbone of the Django site work on the Coastaldefender game code can begin in earnest. However at this point due to the manner in which Django works it is clear that some changes will be necessary within the existing code to make it run more efficiently with Django.

8.7 6th Cycle

8.7.1 Overview

Currently the code is predominately on the client side and does not take advantage of Djangos native abilities. In addition the code is inefficient and verbose, with many of the variables used in the functions hardcoded for a 10 by 10 grid. Ideally the Coastaldefender game would have the variables inherited from a global holder object, this would lend to a more dynamic game code, where a minimal portion of the variables are hard coded but inherited from a parent source.

Once completed the re-factoring of the code will give an accurate portrayal of the final design of the program. A new and improved UML model can be derived from it.

Django uses a MVT (model, view, template) pattern. Originally the majority of the functions and objects relating to the Coastaldefender game were created before the decision to use Django as a content delivery platform had been made, and thus the code is not optimised for Django. With a combination of JQuery, Djangos back end and python scripts the re-factoring of the code will make the following overhauls.

1. To move all of the game logic operations to the server side. This will be used to remove any inconsistencies with various internet browsers, by ensuring the data transferred to it is processed in exactly the same way each time, each session with each user.

2. The view will be handled solely at the client side. To reduce any processes undertaken by the client the only changes to the view will be undertaken.

3. JSON will be used as the data transference medium for sending requests for changes in the map to the server, and the amended data back to the client. JSON was chosen over XML the other most common data

exchange format for its comprehensive integration with both JQuery and Python.

4. For the data interchange JQuerys incredibly powerful GET and POST functions will be used for the Ajax.

8.7.2 Objectives

This mission has two primary objectives, and has no secondary objectives.

1. **Re-factor front end:** To re-factor the code for the front end to handle the Global holder object, the view and the POST and GET functions.

2. **Re-factor Django back end**: To re-factor the back end code to handle the models, the URL handling and the game logic.

8.7.3 The Global Holder Object

The concept of the global holder object is to place all variables that would otherwise be global variables into a single object wrapper, this is for the convenience of having everything in the same same place when debugging, reading and amending the code. More importantly by placing everything in a global object the issue of clashing variables is largely mitigated. It follows the conventions put forward by Douglas Crockford in the creation of the YAHOO global object. The global object is declared in capitals, and the attributes are described in object literals.

```
1  var COASTALDEFENDER = {
2    g : 10,
3    i : 10,
4    row : 0,
5    d : 2*250*Math.cos(Math.PI/6),
6    point : new google.maps.LatLng(52.9757800681602,
         0.6778907775878906),
7    myOptions : {
8      zoom: 12,
9      center: new google.maps.LatLng(52.9757800681602,
           0.6778907775878906),
10     mapTypeControl: true,
11     mapTypeControlOptions: {style: google.maps.
           MapTypeControlStyle.DROPDOWN_MENU},
12     navigationControl: true,
```

```
13      mapTypeId: google.maps.MapTypeId.HYBRID
14    },
15    JSON : null,
16    lineData : {"lineArray":[]},
17    lineExists : false
18  }
```

`g`: This integer is used to determine the number of columns in the hex grid.
`i`: This integer is used to determine the number of rows in the hex grid.
`row`: This integer is the initial value of row in the `gridBuilder` function.
`d`: This equation is the measure of distance used in determining the space between the centre point of adjacent hexes. This is used by the `V3_eshape.js` functions to build the hex polygons.
`point`: This `latlng` coordinate is The initial starting point for the hex grid. All hexes created when the `gridBuilder` function is called will be displaced from this initial latitude and longitudinal coordinate.
`myOptions`: These are the initial options for the creation of the map. They include the centre point for the map, its presented style (in this case satellite image), and its zoom level.
`JSON`: this is an initially null attribute but it becomes populated with a full `JSON` array of the `hexGrid` and its objects via the initial `GET` function. It will be then be further updated via the `POST` function.
`lineData`: This `JSON` object with an empty array is populated by the `buildLine` function called by the `lineCreator` listener attached to each `NewHex` object. With each `NewHex` object designated to have a line built in it the details are appended to empty array in this `JSON` object.
`lineExists`: This Boolean is set to false and is used to in the `buildLine` function in line creation.

8.7.4 Grid Creation

The `hexgrid` construction retains the same principles, though the code has been made shorter and more dynamic in nature. The whole grid creation process is invoked by `newGrid` variable on the `mappage.html` template, which calls the constructor of `HexGrid`.

```
1  function HexGrid () {
2    this.grid = buildGrid();
3  }
```

The `HexGrid` construction calls the `buildGrid` function as a attribute of its instance, ensuring that this is a persistent object.

```
function buildGrid(){
  var grid = new Array(COASTALDEFENDER.g);
  gridPoint = COASTALDEFENDER.point;
  for(var x = 0; x<COASTALDEFENDER.g;x++){
    grid[x] = new Array(COASTALDEFENDER.i);
    gridPoint = displacement(x,gridPoint);
    for(var y = 0; y<COASTALDEFENDER.i;y++){
      var gridHex = new NewHex(x,y,gridPoint);
      grid[x][y] = gridHex;
    }
  }
  return grid;
}
```

The `buildGrid` function performs the same process that `buildGrid` and `buildRow` functions used to perform in previous code iterations. It uses a 2D for loop (using values from the `COASTALDEFENDER` global object as variables). With each new row it calls the displacement function to change the `gridPoint` variable, which displaces teh next row of `NewHex` objects allowing for correct orientation of the `hexGrid`. In addition this each instance of the grid calls a `NewHex` object, which is largely identical to that found in the original build hex grid section of project cycles.

```
function displacement(xCord,gridPoint){
  gridPoint = gridPoint;
  if (xCord == 0){
    return gridPoint;
  }
  else if (xCord%2){
    gridPoint = (EOffsetBearing(gridPoint,COASTALDEFENDER.d
        ,210));
    return gridPoint;
  }
  else {
    gridPoint = (EOffsetBearing(gridPoint,COASTALDEFENDER.d
        ,150));
    return gridPoint;
  }
}
```

The displacement function performs a process identical to that found in the original `buildGrid` function. That is to determine if the row is

odd or even and displace the new row of `NewHex` objects according to the `EOffsetBearing` function found in `eShapes.js`.

The `getJSON` function is part of the JQuery api, it is a compact and excellent way to perform a `GET` Ajax call to the server. The initial function is typical JQuery `(document).ready`, where a function or list of methods is called on the completed loading of the HTML document. In this case it calls the `getJSON` function. The `getJSON` function performs a `GET` from the server retrieving data in the form of a JSON object, in this case it accesses the Django view mapped to the mapmaker URL which creates the JSON server side. On successful callback the JSON is used as an argument in the `mapViewCreator` function.

```
$(document).ready(function(hexGrid){
  $.getJSON("mapmaker/",function(data) {
    mapViewCreator(data);
  });
});
```

`Views.py`

```
def mapmaker(request):
    rawMap = createMap.encodedGrid
    return HttpResponse(rawMap)
  The mapmaker view simply invokes the encodedGrid variable
     from the createMap model and returns it as an
     HttpResponse object.
```

`Models.py`

```
class createMap(models.Model):
    xAxis = 10
    yAxis = 10


    def buildMap(xAxis,yAxis):
        for x in range(0,xAxis):
            theGrid["hexGrid"].append([])
            for y in range(0,yAxis):
                theGrid["hexGrid"][x].append({
                    "xCord":x,
                    "yCord":y,
                    "terrain":startMap[x][y],
                    "modified":0,
                    "hasLine":0
                    })
```

```
17          jsonGrid = json.dumps(theGrid)
18          return jsonGrid
19
20
21      encodedGrid = buildMap(xAxis,yAxis)
```

The `createMap` class contains the `buildMap` function which creates the 2D JSON array of JSON objects. It does this by using a set of 2D for loops and with each instance creating a single JSON object and appending it to the array. It then converts it to a string for transfer back to the client side.

```
1  function mapViewCreator(data){
2    COASTALDEFENDER.JSON = data;
3    jsonGrid =  JSON.stringify(COASTALDEFENDER.JSON);
4    COASTALDEFENDER.JSON = jsonGrid;
5    alert(jsonGrid);
6    for(var x = 0; x<10;x++){
7      for(var y = 0; y<10;y++){
8        newGrid.grid[x][y].terrain = data.hexGrid[x][y].terrain
             ;
9        newGrid.grid[x][y].updateOptions();
10     }
11   }
12 }
```

The successful return function `mapViewCreator` uses the returned JSON from the `GET`. It firstly mutates the `COSTALDEFENDER.JSON` attribute to the contents of the JSON array, and then mutates the `hexGrid` view object terrain attribute to that of the returned JSON array terrain attribute. It finally calls the `updateOptions` method to change the appearance of each hex in the `hexGrid`.

8.7.5 Updating the map

To update the map the process will be to create a JSON array from the list of `NewHex` objects with a designated line. This is then sent as an argument in along with the `COASTALDEFENDER.JSON` attribute so both sets of JSON can be merged server side. Once sent to the correct Django view, the JSON is merged then augmented via various functions server side. The Django view then returns it as an `HttpResponse` object. Once returned the `hexGrid` view object is then mutated to match the returning JSON array. It should be noted that the process of building the line is almost identical to that of

previous iterations, the only difference being that the information regarding to the x coordinate, the y coordinate and the `hasLine` value of each hex selected is appended to the empty array in the `COASTALDEFENDER.lineData` JSON object.

```
$(document).ready(function(){
  $("#turnButton").click(function(){
    alert(COASTALDEFENDER.lineData);
    jsonLine =  JSON.stringify(COASTALDEFENDER.lineData);
    $.post("newTurn/",{"jsonData" :COASTALDEFENDER.JSON, "
       lineData":jsonLine},function(data){
      mapViewUpdater(data);
    }, "json");
  });
});
```

The POST is called from a JQuery function that is bound to the `turnButton` button on the `mappage.html` template. Once called it converts the `LineData` into a string to be sent as an argument, this may seem counter intuitive but without converting to a string the JSON is incorrectly parsed on the server side. The function then calls the POST, which sends the data to the Django view mapped to the `turnButton` url request and sends a JSON object with both the `COASTALDEFENDER.lineData` object and the `COASTALDEFENDER.JSON` object as arguments. Once a successful callback has been made it calls the `mapViewUpdater` function to complete the process.

`views.py`

```
@csrf_exempt
def newTurn(request):
    POSTMap = json.loads(request.POST["jsonData"])
    lineData = json.loads(request.POST["lineData"])
    POSTMap = mapUpdater.mergeData(POSTMap,lineData)
    x = len(POSTMap["hexGrid"][0])
    y = x
    POSTMap = mapUpdater.seaPower(POSTMap,x,y)
    POSTMap = mapUpdater.changeGrid(POSTMap,x,y)
    POSTMap = json.dumps(POSTMap)
    return HttpResponse(POSTMap)
```

The `newTurn` view is csrf exempt. Django has in built csrf protection and the view decorator overrides this, this method was chosen instead of sending appropriate authentication keys since no personal or sensitive data is sent via this process. In turn the view performs the following processes.

1. Both the strings of the `JsonData` and `LineData` are parsed into Json via the `json.loads` method.

2. These strings are then merged together so that the value of `hasLine` in the `jsonData` is converted to that of the value `hasLine` in the `LineData`.

3. (the x and y values) are determined through the size of the `JsonData` array. Please note this currently assumes a square.

4. It calls the `seaPower` function to determine which members of the JSON array are to be mutated.

5. It calls the `changeGrid` function to apply these changes.

6. It converts the JSON to a string and returns it as an `HttpResponse` object.

```
1 def mergeData(JsonData,lineData):
2     z = len(lineData['lineArray'])
3     for i in range(0,z):
4         x = lineData['lineArray'][i]['x']
5         y = lineData['lineArray'][i]['y']
6         JsonData['hexGrid'][x][y]['hasLine'] = lineData['
              lineArray'][i]['hasLine']
7     return JsonData
```

The `mergeData` function is relatively simple. It uses a for loop to cycle through each element the array in the `lineData` JSON object. It then binds variables to its x and y values and mutates the `hasLine` value of the corresponding `jsonData hexGrid` array member to the value of the `hasLine` attribute of the current `lineData` array.

```
1 def seaPower(gridData,x,y):
2     x = x-1
3     y = y-1
4     for i in range(1,x):
5         for j in range(1,y):
6             if gridData['hexGrid'][i][j]['terrain'] == 't'
                  and gridData['hexGrid'][i][j]['hasLine'] == 0
                  :
7                 if checkAjacent(gridData,i,j,'terrain','m'):
8                     gridData['hexGrid'][i][j]['modified'] = 1
9     return gridData
```

The `seaPower` function uses a 2D for loop to cycle through each member of the `JsonData` `hexGrid` array, the limits of the 2D array start at 1 and end at the x and y values -1, meaning that the hexes on the edge of the map are not checked. This is to remove issues of out of bound erros when calling the `checkAjacent` function. It checks to see if the terrain is terrestrial (`t`) and that its `hasLine` value is equal to 0. If this returns true then the hex is then tested via the `checkAjacent` function, and if that returns true the value of the hexs modified attribute is mutated to 1. In the `changeGrid` function any hexes with a modified value of 1 will be mutated further.

```
def checkAjacent(gridData,x,y,attribute,target):
    x = x
    y = y
    isAjacent = False
    for i in range(-1,1):
        for j in range(-1,1):
            if gridData['hexGrid'][i+x][j+y][attribute] ==
               target:
                isAjacent = True
                return isAjacent
```

The `checkAjacent` function uses a 2D for loop to check the values of each hex surrounding the hex being tested. If the attribute argument matches the target argument (in this case the terrain equalling `m`) the function returns true.

```
def changeGrid(gridData,x,y):
    x = x
    y = y
    for i in range(0,x):
        for j in range(0,y):
            if gridData['hexGrid'][i][j]['modified'] == 1:
                gridData['hexGrid'][i][j]['terrain'] = 'm'
                gridData['hexGrid'][i][j]['modified'] = 0
            if gridData['hexGrid'][i][j]['hasLine'] == 1:
                gridData['hexGrid'][i][j]['hasLine'] = 0
    return gridData
```

The `changeGrid` function uses yet another 2D for loop to cycle through each member of the array and then mutates the values of modified, terrain, and `hasLine` if the conditionals return true.

```
function mapViewUpdater(data){
  var response = JSON.stringify(data);
```

```
3   alert(response);
4   for(var x=0; x<10; x++){
5     for(var y=0; y<10; y++){
6       newGrid.grid[x][y].xCord = data.hexGrid[x][y].xCord;
7       newGrid.grid[x][y].yCord = data.hexGrid[x][y].yCord;
8       newGrid.grid[x][y].terrain = data.hexGrid[x][y].terrain
            ;
9       newGrid.grid[x][y].updateOptions();
10      newGrid.grid[x][y].hasLine = false;
11    }
12  }
13  COASTALDEFENDER.JSON = data;
14  jsonGrid =  JSON.stringify(COASTALDEFENDER.JSON);
15  COASTALDEFENDER.JSON = jsonGrid;
16  COASTALDEFENDER.lineData = {"lineArray":[]};
17  COASTALDEFENDER.lineExists = false;
18  theLine.setMap(null);
19 }
```

The `mapviewUpdater` is called when a successful `POST` callback is made. It uses the `POST` data returned as the only argument. It then uses a 2D for loop to access each member view `HexGrid` and mutates the values of terrain and has line to that matching the returned JSON array. It also calls the `updateOptions` method which changes the appearance of the hex in accordance to the terrain value.

Once each hex has been mutated the function does some general house cleaning. It updates the value of `COASTALDEFENDER.JSON` to that of the returned JSON array and converts it to a string to avoid parsing issues in future `POST` calls. In addition it resets `lineData` and line exists to their default values as well as removing `theLine` object from the map.

8.7.6 Summary

The process of re-factoring has made a much more robust and "proper" set of code. It uses more technologies such as Django and JQuery to their potential and resolves many previous issues that emerged in the code from the early part of the project. However after completing this portion of re-factoring there are still a number of steps that could be undertaken to further improve the code.

The initial `GET` could be converted into a `POST` and the JSON array could be originally made client side. This may seem to go against the philosophical

concepts of performing as much as possible on the server, but it does mean that all of the variables that are used in the game can be stored within the global object. This would make the game very easy to modify and personalise.

8.8 7th Cycle

8.8.1 Overview

So far all the mission work has been based around creating a game which builds a model of coastal erosion, and allows a player to interact with it. However this game seeks to have the greater aim of stimulating conversation around coastal management by co-opting social media to facilitate interaction. Initially this project will be looking at Twitter as the primary social media platform, since it almost ubiquitous in use and designed for rapid conveyance of information in small bytes.

8.8.2 Overview

Compared to previous iterations of the the project cycles implementing Twitter into the game should be a relatively simple process. However the functionality of the integration should perform two actions; firstly to send a status update to the social media associated with the Coastal Defender Game, secondly to dynamically generate content for the status update that reflects the current game state.

This mission has two primary objectives, and no secondary objectives.

1. **Game State to Tweet:** To create the code that will send the tweet along with the content that will reflect the current game state.
2. **Tweet to Game State:** To create a way that a tweet containing the game state can create a version of the Coastal Defender game reflecting its contents.

8.8.3 The Twitter Button

The Twitter API contains simple pre-generated code for creating a button on the web page. It uses simple HTML and some proprietary functions to achieve this.

```
1 <a href="https://twitter.com/share" class="twitter-share-
    button" data-text="hi">Tweet</a>
2 <script>!function(d,s,id){var js,fjs=d.getElementsByTagName(s
    )[0];if(!d.getElementById(id)){js=d.createElement(s);js.id
    =id;js.src="//platform.twitter.com/widgets.js";fjs.
    parentNode.insertBefore(js,fjs);}}(document,"script","
    twitter-wjs");</script>
```

The above code is generated by the Twitter API for the integration of the Tweet button.

8.8.4 Updating the Tweet with the Game State

The content of the Tweet is defined by the HTML code data-text this can be augmented though use of JQuerys `attr` function. This would be called as the final method in the `newTurn` POST JQuery function

```
1 $(   #tweetbutton ).attr('data-text', COASTALDEFENDER.
    gameStateString);
```

The `attr` function targets a DOM element (in this case the `tweetButton`) and replaces the contents defined by the first argument by the data in the second argument. In this case it replaces the string found in data-text by the string found in `COASTALDEFENDER.gameStateString`.

This works effectively, however two complications arise. The first is the limitation of the string length for tweets, its famous 140 character limit. The game state string sent to the tweet contains the terrain character of that hex (e.g. t or m), and whilst this works for the test grid which stands at a ten by ten, a final version of the game would have a much higher number far exceeding the 140 character limit. Overcoming this obstacle will be a major part of the project, and will determine how the player interacts with the game state.

The second issue is much more immediate, whilst the JQuery `attr` function works the first time, each subsequent call of the `newTurn` button will not change the content of the data-text field. This is due to the manner that the twitter API constructs the Iframes around the button.

To resolve the asynchronous API issue, the Twitter developers have created the `twttr.widgets.load()` function that re-invokes the twitter button and functions. Used in conjunction with code to rebuild the Twitter Iframes

around the tweet button this allows for asynchronous reloading of the button and refreshing the contents of data-text.

```
1 $('#tweetButton iframe').remove();
2         var tweetButton = $('<a></a>')
3           .addClass('twitter-share-button')
4           .attr('href', 'http://twitter.com/share')
5           .attr('data-text', COASTALDEFENDER.gameStateString)
              ;
6         $('#tweetButton').append(tweetButton);
7         twttr.widgets.load();
```

The above code was sourced from jsFiddle (`http://jsfiddle.net/45ddY/`) and was created by Chris Francis. It deletes the existing Tweet Button and then rebuilds it, setting the contents of `COASTALDEFENDER.gameStateString` as its data-text string. Once rebuilt it then calls the `twttr.widgets.load();` to re-invoke the button.

8.8.5 Summary

Implementing the Twitter button was relatively simple. However its final appearance and use will all depend on this how this project will display a game state. The next part of the project cycles will determine this.

8.9 8th Cycle

8.9.1 Overview

A fundamental part of the the game's design is to be able to save and retrieve previously saved game states. This will allow the players to share current game states with others who can then retrieve it from the database to continue that game or critique their choices.

Django uses an SQL database to store all of the objects within the website, be this a blog post or in the case of Coastaldefender the frame for the map into which the game is played. All of these objects can be viewed through Django's admin page, and if the model for the object has the appropriate attributes text, images and hyper links can be added. An example of the Django admin manual entry can be seen in.

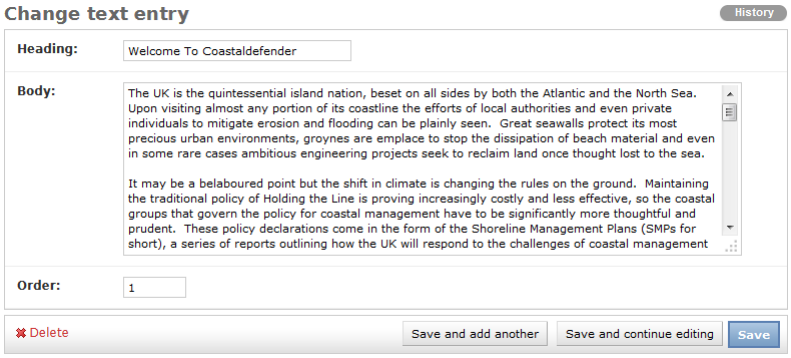

Figure 82: An example of a completed Django Admin form.

The aim of this specific mission will be to create an object within the Django database which can then be viewed within the Django admin. Usually this is done through the invocation of models via the models.py file, but in this case a new model can be saved if correctly implemented by using the `.save()` method.

8.9.2 Objectives

This mission only has a single primary objective, and no secondary objectives.
Create and save
GameState: Create a function that will save the game state into the Django database.

8.9.3 Save method

Creating the game object will attempt to take the current game state after augmentations have been made to reflect the model of seapower and change. It will be later converted to JSON and sent back to the client side, it is therefore currently a Python dictionary

Creating a new game state as a Django database object is. Firstly an empty model in models.py must be created so that multiple versions of it can be saved to the database later.
models.py

```
1 class gameState (models.Model):
2     terrainString = models.TextField()
```

This game model has a single attribute of `terrainString` that will be used to store the current game state as a string. The `saveState()` function is called in views.py to create a version of the `gameState` model

views.py

```
turnId = mapUpdater.saveState(POSTMap)

mapupdater.py
def saveState(gameData):
        turnState = gameState(terrainString = gameData)
        turnState.save()
        turnId = turnState.id
        return turnId
```

This creates an variable that is saved as a new object within the database. The `saveState` function itself creates a new version of the `gameState` object and applies the current game state to it's `terrainString` attribute (in this case this has been passed as the `gameData` argument). The `save()` method is then called to make this new version of the `gameState` object a persistent object within the database.

8.9.4 Creating a unique save sate

However steps need to be taken to ensure that each entry into the database is unique. So that no two possible layouts of hexes within the game are repeated in the database. To achieve this the `saveState()` function is updated.

mapUpdater.py

```
def saveState(gameData):
    try:
        turnState = gameState.objects.get(terrainString =
            gameData)
    except:
        turnState = gameState(terrainString = gameData)
        turnState.save()
    finally:
        turnId = turnState.id
        return turnId
```

The function now attempts to retrieve an object from the Djagno database where the `terrainString` attribute matches the `gameData` argument. If an exception is raised (in this case it will raise an exception if no matching

object is found), a new object will be created using the method set out in the previous section.

The function returns the object id which is the primary key automatically generated by each new entry of an object into the Django database.

8.9.5 Summary

Creating a save state is a relatively simple process, but it is important to check the database for similar entries. By using the try, except and finally statement the database can be successfully searched and where required a new entry can be made.

To make a successful save game however the history of the game states thus far in the game instance needs to be recorded too. In future iterations of the the save game system using the `saveState` object id can be used to record the game state history for that instance of the game thus far.

8.10 9th Cycle

8.10.1 Overview

To make the game interesting, unpredictable and more representative of an actual game model functions need to be inserted into the server side code to create variable outcomes. This will be achieved though augmenting the seapower function by creating a number of variables which will effect its returned value.

Currently the seapower function only checks each hex to see if it is a terrestrial hex and if it is then adjacent to a marine hex. If true it then automatically converts the hex into another marine environment. To iterate on the complexity the following objectives need to be achieved. Create multiple types of terrestrial type hexes, each of which will have specific resistance against seapower. This information needs to be stored in the `COASTALDEFENDER` object since it is a potentially mutable game variable.

Exposed areas of the land need to be more susceptible to seapower. This is to represent the additional battering by the sea that places structures like headlands receive.

The implementation of the Bias mesh, so that a second map of meta data can be used to tweak the seapower strength from hex to hex. Again this would be stored in the `COASTALDEFENDER` object.

Finally a new function needs to be created to reclaim marine hexes and convert their environment into terrestrial, when the outcome returned by the seapower function demands it.

8.10.2 Objectives

This mission has three primary objectives.
Create realistic Seapower: create a "realistic" model of coastal erosion and change.
Create bias Mesh: create an overlay which can effect the way in which the seapower works
Integrate bias Mesh: emplace a way to use the bias mesh to effect the seapower function
seapower reclaim land: create a function to turn marine into terrestrial environments if the seapower function demands it.

8.10.3 Environment Types

The number of environments within the game have been expanded upon, the finished map will appear more varied than previous iterations appearing within the game map as seen in figure.

Figure 83: An example map layout showing the wider range of environments available.

This is achieved by including a greater range of possible environment types found in `COASTALDEFENDER` object. The new version of the example start map reflects this
Mapfunctions.js

```
1 startMap : [
2 ["n","n","n","n","n","n","n","n","n","n","n"],
```

```
["n","m","m","m","m","m","m","m","m","n"],
["n","m","m","m","m","m","m","m","m","n"],
["n","m","m","m","m","m","m","m","m","n"],
["n","m","m","m","m","m","m","m","m","n"],
["n","b","b","b","b","b","b","b","b","n"],
["n","d","d","d","d","d","d","d","d","n"],
["n","d","hf","hf","hf","hf","sm","sm","d","n"],
["n","d","hf","hu","hu","hf","sm","sm","d","n"],
["n","n","n","n","n","n","n","n","n","n"],
],
```

```
if (this.terrain == "hu")
    fillColour = "#666666";
  this.hex.setOptions({fillColor:fillColour});
```

The greater range of colours was achieved by adding additional fill colours for the `updateOptions` method attached to each hex.

```
resistList : {
    'n' : 0,
    'm' : 0,
    'b' : 10,
    'd' : 12,
    'sm' : 15,
    'hf' : 17,
    'hu' : 19,
    }
```

The resist list is a set of values that represents the variable of resistance of that specific hex to be used in the `seaPower` function. As of such this needs to be passed as part of the JSON object to the server side as part of the `newTurnButton` POST function.

On the server side views.py sends passes the relevant arguments to the seapower function.

8.10.4 Bias Map

Mapfunctions.js

```
biasMap : [
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,-1,-1,0,0,0,0],
[0,0,0,-1,-1,-1,-1,0,0,0],
```

```
[0,0,-1,-1,-1,-1,-1,-1,0,0],
[0,-1,-1,-1,-1,-1,-1,-1,-1,0],
[0,-1,-1,-1,-1,-1,-1,-1,-1,0],
[0,-1,2,2,2,2,0,0,-1,0],
[0,-1,2,3,3,2,0,0,-1,0],
[0,0,0,0,0,0,0,0,0,0],
],
```

The bias maps is implemented by the creation of an additional 2D array of integer values as an attribute of the `COASTALDEFENDER` object. The Bias map mirrors the dimensions of `startMap`. The bias map is additionally passed to the server as part of the JSON object. On the server side this is passed as an argument to the `seaPower` function through views.py

8.10.5 Modelling Exposure and the Seapower Function

mapUpdater.py

```
def seaPower(gridData,x,y,resistList,mapBias):
    x = x
    y = y
    for i in range(1,x-1):
        for j in range(1,y-1):
            if (gridData['hexGrid'][i][j]['terrain'] != 'm'):
                ajacentCount = checkAjacent(gridData,i,j,'
                   terrain','m')
                if ajacentCount >= 1:
                    resistKey = gridData['hexGrid'][i][j]['
                       terrain']
                    resistMod = resistList[resistKey]
                    biasMod = mapBias[i][j]
                    modifiedTo = seaAttack(gridData['hexGrid'
                       ][i][j]['terrain'],ajacentCount,
                       resistMod,biasMod)
                    gridData['hexGrid'][i][j]['modified'] =
                       modifiedTo
    return gridData
```

The `seaPower` function takes the current game map as a variable `(gridData)`, the map dimensions `(x,y)`, the resist list and the bias map as arguments. Firstly the function checks each each hex via a 2D for loop to see if that hex has a terrestrial environment (in this case not `'m'`). If this is the case it calls the `checkAjacent` function which now instead of returning a boolean returns an integer value equal to the number of adjacent marine hexes.

If the hex has `ajacentCount` above zero it then calls the `seaAttack` function which will determine the chance of it converting its environment to marine.

```
def seaAttack(gridId,ajacentCount,resistMod,biasMod):
    diffculty = resistMod
    seaAttack = random.randrange(0,20,1)
    bias = biasMod

    ajacentMod = ajacentCount - 2
    if ajacentMod < 0:
        ajacentMod = 0

    vsSea = (diffculty + bias) - (seaAttack + ajacentMod)

    if vsSea <= 0:
        return 'lost'
    elif vsSea in range(1,10):
        return 'remain'
    elif vsSea >= 11:
        return 'gain'
```

the `seaAttack` function takes the current hex environment attribute, its correlating resist value (from the `resistList`). It also takes the value returned by call adjacent (`ajacentCount`), as well as the corresponding value for the bias map (`biasMod`).

Firstly it creates a variable called `SeaAttack` which is a random integer between 0-20. It then augments the `ajacentCount` value to represent something less skewing to the result.

The resulting `vsSea` integer variable is the combined result of the resist value of the hex's environment plus the bias map value, from which it is misused. The combined value of the `seaAttack` random variable plus the augmented Adjacent mod.

Depending on the `vsSea` value it will return `lost`, `remain` or `gain`. Where `lost` means the hex will be converted into marine environment in the `changeGrid` function, `remain` means no change and `gain` will call the `reclaimLand` function during the `changeGrid` function.

```
def reclaimLand(gridData,x,y):
    x = x
    y = y
    for i in range(-1,1):
        for j in range(-1,1):
```

```
6                 if gridData['hexGrid'][i+x][j+y]['terrain'] == 'm
                      ':
7                     gridData['hexGrid'][i+x][j+y]['terrain'] = 'b
                          '
```

`ReclaimLand` checks each adjacent hex's environment and if that environment is marine it converts it to a beach environment.

8.10.6 Summary

Whilst not overly complex the `seaPower` function provides enough variability to create an unsure out come. True modelling of coastal processes would be maddeningly complex and also outside the scope of this project.

When it comes to line designation which will iterate the complexity somewhat many of the same techniques used her can apply. Also where possible as much of the variables should be moved the the `COASTALDEFENDER` object as possible.

8.11 10th Cycle

8.11.1 Overview

This mission aims to update the manner in which the lines are designated to each hex. So that multiple designation types can be made.

The system of designating the lines to the hex grid requires greater flexibility, also the appearance of the line literally as a drawn line provides a few problematic elements due to the way Google Maps draws polylines. A more effective method of designating lines would be to designate a hex and to change the hex's appearance. The advantages of this method would be.

1. Greater flexibility as to which hexes can be designated. Due to the manner of how poly lines are drawn they have to currently be adjacent to one another. Whilst this issue may not be too problematic with just a single line designation type it becomes an issue when designating multiple types and therefore multiple lines.

2. Visual representation is much more obvious for the player. By changing the colour of the hex is it easily identifiable to the player as to which hex is designated as which line type.

8.11.2 Objectives

This mission has four primary objectives and no secondary objectives.
complex lines view: update the system of line creation to allow for multiple line types
complex lines server side: update the functions that deal with seapower and incorporate line modifiers
resource counter: create a UI object within the Google Maps interface that will count the amount of resources the player has to create lines.
turn counter: create a UI object that shows the current turn in the game

8.11.3 Creating a Context Menu for Line Designation

The current method is to build a polyline and to add a new point in its path with each new hex that is designated as a specific line type thus increasing the size of the line with each new designation. The new method will simply change the colour of the hex thus simplifying the process greatly.

To do this a a method for creating the designation options must be created for the player. In this case it will be done by creating a context menu when right clicking on a chosen hex. Context menus are not part of the standard Google Maps API so the ContextMenu.js libary created by Martin Pearman was used (`http://googlemapsmania.blogspot.de/2012/04/create-google-maps-context-men`

The `contextMenu` function is a prototype function for the `NewHex` object constructor. It is invoked at the creation of each `NewHex` during the initial hex grid creation. Additionally to facilitate this a `contextMenuOptions` global object is created for each instance of the context menu to access..
mapScreen.html

```
1 var contextMenuOptions = {};
2   contextMenuOptions.classNames = {menu:'context_menu',
       menuSeparator:'context_menu_separator'};
3
4   var menuItems = [];
5   menuItems.push({className:'context_menu_item', eventName:'
       hold_the_line', label:'Hold The Line'});
6   menuItems.push({});
7   menuItems.push({className:'context_menu_item', eventName:'
       advance_the_line', label:'Advance The Line'});
8   menuItems.push({});
9   menuItems.push({className:'context_menu_item', eventName:'
       managed_realignment', label:'Managed Realignment'});
```

```
10    menuItems.push({});
11    menuItems.push({className:'context_menu_item', eventName:'
          cancel_assignment', label:'Cancel Assignment'});
12    contextMenuOptions.menuItems = menuItems;
```

mapfunctions.js

```
1  NewHex.prototype.createMenu = function (hexShape,lineDetails,
       x,y){
2    this.menuList = google.maps.event.addListener(this.
         contextMenu, 'menu_item_selected', function(latLng,
         eventName){
3      if(lineDetails == "NAI"){
4        if (eventName == "hold_the_line"){
5          if(COASTALDEFENDER.currentResources - COASTALDEFENDER
               .lineCost.HTL <= 0){
6            alert("You do not have enough resources to assign
                 this policy to the hex.");
7          }
8          else{
9            lineDetails = "HTL";
10           newGrid.grid[x][y].hasLine = lineDetails;
11           COASTALDEFENDER.currentResources = COASTALDEFENDER.
                 currentResources - COASTALDEFENDER.lineCost.HTL;
12           hexShape.setOptions({strokeColor : COASTALDEFENDER.
                 colourList.HTLLine, strokeWeight : 6});
13           alert("" + COASTALDEFENDER.currentResources + "," +
                  newGrid.grid[x][y].hasLine + "," +
                 COASTALDEFENDER.colourList["HTLLine"]);
14           document.getElementById("over_map_right").innerHTML
                  = "Resources available:" + COASTALDEFENDER.
                 currentResources;
15         }
16
17
18       }
19
20       if (eventName == "advance_the_line"){
21         if(COASTALDEFENDER.currentResources - COASTALDEFENDER
               .lineCost.HTL <= 0){
22           alert("You do not have enough resources to assign
                 this policy to the hex.");
23         }
24         else{
25         lineDetails = "ATL";
26         newGrid.grid[x][y].hasLine = lineDetails;
```

```
27          COASTALDEFENDER.currentResources = COASTALDEFENDER.
                currentResources - COASTALDEFENDER.lineCost.ATL;
28          hexShape.setOptions({strokeColor : COASTALDEFENDER.
                colourList.ATLLine, strokeWeight : 6});
29          document.getElementById("over_map_right").innerHTML =
                 "Resources available:" + COASTALDEFENDER.
                currentResources;
30          }
31        }
32
33        if (eventName == "managed_realignment"){
34          if(COASTALDEFENDER.currentResources - COASTALDEFENDER
                .lineCost.HTL <= 0){
35            alert("You do not have enough resources to assign
                   this policy to the hex.");
36          }
37          else{
38          lineDetails = "MR";
39          newGrid.grid[x][y].hasLine = lineDetails;
40          COASTALDEFENDER.currentResources = COASTALDEFENDER.
                currentResources - COASTALDEFENDER.lineCost.MR;
41          hexShape.setOptions({strokeColor : COASTALDEFENDER.
                colourList.MRLine, strokeWeight : 6});
42          document.getElementById("over_map_right").innerHTML =
                 "Resources available:" + COASTALDEFENDER.
                currentResources;
43          }
44        }
45      }
46      else{
47        if (eventName == "managed_realignment" || eventName ==
             "advance_the_line" || eventName == "hold_the_line"){
48          alert("This hex has already been assigned.");
49        }
50        if(eventName == "cancel_assignment"){
51          lineType = newGrid.grid[x][y].hasLine;
52          if (lineType = "HTL"){
53            lineValue = COASTALDEFENDER.lineCost.HTL;
54          }
55          if (lineType = "ATL"){
56            lineValue = COASTALDEFENDER.lineCost.ATL;
57          }
58          if (lineType = "MR"){
59            lineValue = COASTALDEFENDER.lineCost.MR;
60          }
```

```
61          COASTALDEFENDER.currentResources = COASTALDEFENDER.
                currentResources + lineValue;
62          lineDetails = "NAI";
63          newGrid.grid[x][y].hasLine = lineDetails;
64          hexShape.setOptions({strokeColor : COASTALDEFENDER.
                colourList.nullHex, strokeWeight : 0.5});
65          document.getElementById("over_map_right").innerHTML =
                 "Resources available:" + COASTALDEFENDER.
                currentResources;
66        }
67      }
68    });
69  }
```

The `ContextMenu` function creates a listener for each hex that is called when a right click is made on that hex. It then uses the ContextMenu.js to draw the context menu on the map. The rest of the function is a series of logic conditions that tests to see if the player currently has enough resources, if a line is already present and to which type of line has been designated for that hex. There is also an option to cancel the assignment thus refunding the resources.

By designating a hex with a chosen line type the `COASTALDEFENDER.currentResources` attribute will be updated with the appropriate resource deduction which is determined by the `COASTALDER.lineCost` attribute list. The function then updates the colour of the hex to that of line designation which is stored in the `COASTALDEENDER.colourList` attribute.

Updating the `updateOptions` to change hex colours To create a more dynamic function for colour setting on induvual hexes the Update options will get its colour options form a dictionary within the `COSTALDEFENDER.colourList` attribute. When calling the `setOptions` method in the `contextMenu` function it also determines the colour from this attribute.

8.11.4 Creating the Resource Counter and Turn Counter

The resource counter will display the current value of the `COSTALDEFENDER.currentResources`. It is a simple Div in page template.

mapScreen.html

```
1  <div id="wrapper">
2    <div id="map_canvas" style="width:100%; height:100%"></div>
3    <div id="over_map">Current Turn: 1</div>
4    <div id="over_map_right">Resources available: 10 </div>
```

```
5 </div>
```

Updating the Div content is done via the `contextMenu` function via a `getElementById` and using the `InnerHTML` method to update its content to that of Div.

At the beginning of each turn (when the POST function returns) the `COSTALDEFENDER.currentResources` is reset to its base amount.

MapFunctions.js

```
1   COASTALDEFENDER.currentTurn = COASTALDEFENDER.currentTurn +
        1;
2   COASTALDEFENDER.currentResources = COASTALDEFENDER.
        maxResources;
3   $("#over_map").text("Current Turn:" + COASTALDEFENDER.
        currentTurn);
```

The turn counter is a much easier process, with the current turn value being updated at the beginning of each new turn.

8.11.5 Server side additions

Small changes to the server side have been made to take into account for the addition of line types that will provide an additional modifier for the `seaAttack` function.

mapUpdater.py

```
1  def seaAttack(ajacentCount,resistMod,biasMod,lineMod):
2      diffculty = resistMod
3      seaAttack = random.randrange(0,20,1)
4      bias = biasMod
5      line = lineMod
6
7      ajacentMod = ajacentCount - 2
8      if ajacentMod < 0:
9          ajacentMod = 0
10
11     vsSea = (diffculty + bias + line) - (seaAttack +
           ajacentMod)
12
13     if vsSea <= 0:
14         return 'lost'
15     elif vsSea in range(1,10):
16         return 'remain'
17     elif vsSea >= 11:
```

```
18          return 'gain'
```

In the `seaAttack` function there is now a `lineMod` argument passed to it which is the value for the current hex's line designation. The `lineMod` value is passed to the server from the POST function as part of the JSON object.

8.11.6 Summary

With the addition of a complex line implementation system Coastaldefender as a working game is now complete. However the issue of reverting to previous game states still needs to be addressed. In addition it would be nice to implement an option so that users can apply the official SMP designations automatically. This would provide additional context.

8.12 11th Cycle

8.12.1 Overview

With the game play complete. The other main feature of the game must be created, which is to be able to load previous game states for a player to refer back to. This will be done via one of two ways. The first will simply be to select a previously generated turn from a turn list. The second method will be to load an entire game history through a game ID.

8.12.2 Objectives

This mission has three primary objectives and no secondary objectives.

develop game state retrieval methodology: To determine a generic method that can be used to retrieve objects from the Django database.
Create load by ID: Using the retrieval create a method to load a game (complete with history).
Create a history viewer: Using the retrieval method create a list of previous turns which the user can select to play.

8.12.3 Game State retrieval Method

The method to retrieve an object from the Django database is relatively simple and has already been demonstrated in the 8th development cycle. By using the Django API one can retrieve an object (or objects) from the database via the `objects.get()` method.

```
currentTurn = gameState.objects.get(id = turnData)
```

In this instance a new variable is generated that is equal to the object returned from the database. The filter for this `.objects.get()` method is id, which is the primary key field for `gameState` class. Every time a new instance of any class is created in the Django database a primary key is generated, this provides an easy solution for the generated game id. When a user tweets the game ID the primary key for that unique `gameStateHistory` object can be used.

8.12.4 Load By Unique Game ID

To retrieve a saved game state from the client side firstly JQuery is used to call the relevant Django view.

mapScreen.html

```
<input type="text" id="loadIdInput" name ="Load Game" value="
    Enter a game ID...">
<input type="submit" id="loadIdSubmit" value="Submit">
```

```
$(document).ready(function(){
  $( "#loadIdSubmit" ).click(function() {
    loadById();
  });
});
```

This generates a button via HTML that is linked to a JQuery listener that will be triggered once the submit button is clicked. This trigger calls the `loadById()` function.

The `loadByid` function is an Ajax function similar to the `loadNewTurn` POST call created in the 3rd development cycle. It sends a JSON object with a single key value pair to the server. This `gameIdData` is equal to the game ID submitted and it calls the `loadSavedGame` Django view.

views.py

```
@csrf_exempt
def loadSavedGame(request):
    POSTId = json.loads(request.POST["gameIdData"])
    retrivedHistory = mapUpdater.loadHistory(POSTId)
    historyData = retrivedHistory.historyString
    currentTurn = mapUpdater.convertHistoryData(historyData)
    retrivedGame = mapUpdater.revertOldTurn(currentTurn)
```

```
8       retrivedGame = retrivedGame.terrainString
9       retrivedGame = json.loads(retrivedGame)
10
11      uniqueId = mapUpdater.getGameId(historyData)
12
13      returnData = {}
14      returnData["gameData"] = retrivedGame
15      returnData["historyData"] = historyData
16      returnData["uniqueId"] = uniqueId
17      returnJSON = json.dumps(returnData)
18      return HttpResponse(returnJSON)
```

The view uses the loads method to convert the JSON into its python analogue. Then retrieves the `gameStateHistory` whos primary key id is equal to the submitted game id through the `loadHistory` function. The `historyData` variable will be the `historyString` attribute of the returned object and will contain the complete game history.

mapUpdater.py

```
1  def loadHistory(gameId):
2      try:
3          responseId = gameStateHistory.objects.get(id = gameId
               )
4      except:
5          responseId = "Unfortuantly this game does not exist"
6      finally:
7          return responseId
```

If it fails to return a valid `gameStateHistory` object it raises an exception text.

To generate the primary key for the current turn of the submitted game ID the view calls the `convertHistory` function will simply returns the final element of the `historyData` array.

```
1  def convertHistoryData(historyData):
2      convertedHistoryData = json.loads(historyData)
3      currentTurn = convertedHistoryData[-1]
4      return currentTurn
```

This returned id is used as an argument in the `reverOldTurn` function to retrieve the `gameState` object of that specific turn. The returned `gameState` is converted into a python dictionary so it can be packaged JSON correctly. Finally the current game ID is retrieved via the same `objects.get` method.

```
def revertOldTurn(turnData):
    currentTurn = gameState.objects.get(id = turnData)
    return currentTurn
```

```
def getGameId(historyData):
    responseId = gameStateHistory.objects.get(historyString =
        historyData)
    responseId = responseId.id
    return responseId
```

Once all variables are correctly formatted they are packaged into a python dictionary and conveyed into JSON to be sent as a Http response object to the client side, where it is dealt with by the `loadById` response function.

MapFunctions.js

```
function loadById(){
  gameIdData = $("#loadIdInput").val();
  $.post("loadSavedGame/",{ "gameIdData" : gameIdData },
     function(data){
    gridData = data.gameData;
    uniqueId = data.uniqueId;
    historyUpdater(data);
    currentTurn = COASTALDEFENDER.gameStateHistory.length;
    COASTALDEFENDER.currentTurn = currentTurn;
    turnUpdater(COASTALDEFENDER.maxResources, uniqueId);
    removeAllTurnList();
    var ajustedTurn = currentTurn -1;
    for(var x=0; x < currentTurn; x++){
      var turn = x + 1;
      var turnId = '#turn' + turn;
      var listItem = new ExpandTurnList(turnId, turn,
         COASTALDEFENDER.gameStateHistory);
      listItem.applyListener(turnId, turn, COASTALDEFENDER.
         gameStateHistory);
    }
    mapViewUpdater(gridData);

    inputData = "Check out my plans for Coastal Defence #
       CoastalDefender #GameID" + COASTALDEFENDER.
       currentGameId;
    //Solution provided by Chirs Francis via jsFiddle http://
       jsfiddle.net/45ddY/
    $('#tweetButton iframe').remove();
      var tweetButton = $('<a></a>')
      .addClass('twitter-share-button')
```

```
25          .attr('href', 'http://twitter.com/share')
26          .attr('data-text', inputData);
27      $('#tweetButton').append(tweetButton);
28      twttr.widgets.load();
29
30    }, "json");
31  }
```

Upon return the `loadById` function begins the process of running the various updater functions. The `historyUpdater` function changes the value of `COASTALDEFENDER.gameStateHistory`. The `turnUpdater` function changes the value of the `COASTALDEFENDER.currentTurn` attribute and applies this value to the appropriate HTML DOM element to be displayed on the game UI.

```
1  function historyUpdater(data){
2    var history = data.historyData;
3    var historyData = $.parseJSON(history);
4    COASTALDEFENDER.gameStateHistory = historyData;
5  }
```

```
1  function turnUpdater(resourceMax, uniqueId){
2    COASTALDEFENDER.currentGameId = uniqueId;
3    COASTALDEFENDER.currentTurn = COASTALDEFENDER.currentTurn +
         1;
4
5    CDtext = typeof COASTALDEFENDER.currentGameId;
6    alert(CDtext);
7    COASTALDEFENDER.currentResources = resourceMax;
8    $("#over_map").text("Current Turn:" + COASTALDEFENDER.
       currentTurn);
9    $("#over_map_right").text("Current Resources:" +
       COASTALDEFENDER.currentResources);
10   $("#over_map_rightB").text("Current Game ID:" +
       COASTALDEFENDER.currentGameId);
11 }
```

The `mapViewUpdater` function is then called to apply the changes to the hex grid to the client side. This has been described in the 3rd development cycle.

Finally it creates a new instance of the `ExpandTurnList` constructor (which will be described in the next section) and redraws the iframes for the tweet button.

8.12.5 Creating a Previous Turn Selector

The other way to load a saved game will be to select it directly from the previous game state list. This list will grow turn by turn. The turn selector will be created as a JavaScript object since each new selector will be required to be persistent.

mapScreen.html

```
<div id="turnMenu">
  <ul class="menu">
    <li>
      <a id="selector" href="#">Turn Selection</a>
      <ul id="revertMeunu" >
      </ul>
    </li>
  </ul>
</div>
```

MapFunctions.js

```
function ExpandTurnList(turnId, displayTurn, currentHistory){
  $("#revertMeunu").append('<li><a href="#" class="
      revertTurnItem" id=' + '"turn' + displayTurn + '">Turn '
       + displayTurn + '</a>');
}
```

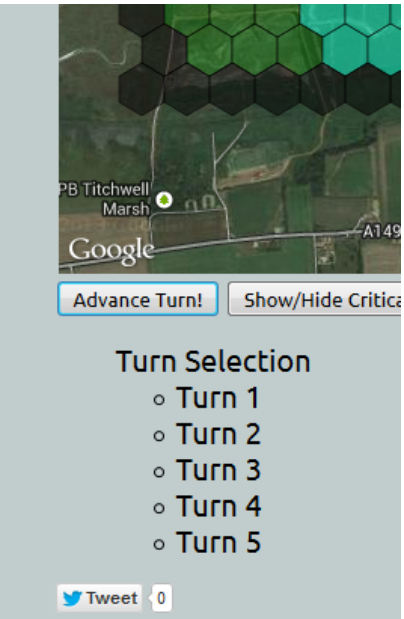

Figure 84: The displayed turn selector list.

Each new `ExpandTurnList` instance will be a simple HTML list li element that is affixed to the `revertMenu` ul list element. It will create a new list entry as thouse show in figure .

```
1  ExpandTurnList.prototype.applyListener = function(turnId,
       displayTurn, currentHistory){
2    $(turnId).on("click", function(){
3      if(displayTurn == 1){
4        alert("This would be reverting back to the first turn,
             click Restart Game instead.");
5      }
6      else{
7        revertToHistory(0, displayTurn);
8        JSONHistory = JSON.stringify(COASTALDEFENDER.
             gameStateHistory);
9        $.post("loadOldTurn/",{"turnData" : currentHistory[
             displayTurn] , "historyData" : JSONHistory },
             function(data){
10         uniqueId = data.uniqueId;
11         gridData = data.gameData;
12         mapViewUpdater(gridData);
13         revertToHistory(0,displayTurn);
```

```
14          revertToTurn(COASTALDEFENDER.maxResources,displayTurn
                , uniqueId);
15          removeTurnList(displayTurn);
16
17          inputData = "Check out my plans for Coastal Defence #
                CoastalDefender #GameID" + COASTALDEFENDER.
                currentGameId;
18          //Solution provided by Chirs Francis via jsFiddle
                http://jsfiddle.net/45ddY/
19          $('#tweetButton iframe').remove();
20            var tweetButton = $('<a></a>')
21            .addClass('twitter-share-button')
22            .attr('href', 'http://twitter.com/share')
23            .attr('data-text', inputData);
24          $('#tweetButton').append(tweetButton);
25          twttr.widgets.load();
26
27        }, "json");
28      }
29    });
30  }
```

The `applyListener` prototype function acts recursively by creating a function that will retrieve the game ID at the point of this Object being created. It calls the `loadOldTurn` view that works almost identically to the `loadSavedGame`. It converts the JSON into its python analogue, calls the `revertOldTurn` function to to return the new game State and finally packages it into a dictionary which is returned back as JSON in a Http response object.

views.py

```
1  @csrf_exempt
2  def loadOldTurn(request):
3      POSTHistory = json.loads(request.POST["historyData"])
4      POSTTurn = json.loads(request.POST["turnData"])
5
6      retrivedTurn = mapUpdater.revertOldTurn(POSTTurn)
7      retrivedTurn = retrivedTurn.terrainString
8      retrivedTurn = json.loads(retrivedTurn)
9
10     uniqueId = mapUpdater.getGameId(POSTHistory)
11
12     returnData = {}
13     returnData["gameData"] = retrivedTurn
14     returnData["uniqueId"] = uniqueId
```

```
15      returnJSON = json.dumps(returnData)
16      return HttpResponse(returnJSON)
```

Upon returning the JSON is used to update the `COASTALDEFENDER` object in a similar manner to the previously described updater functions in section 8.12.4, through the `revertToHistory` and `revertToTurn` functions.

MapFunctions.js

```
1  function revertToHistory(sliceStart, sliceStop){
2    newHistory = COASTALDEFENDER.gameStateHistory.slice(
         sliceStart, sliceStop);
3    COASTALDEFENDER.gameStateHistory = newHistory;
4  }
```

```
1  function revertToTurn(resourceMax, newTurn, uniqueId){
2    COASTALDEFENDER.currentGameId = uniqueId;
3    COASTALDEFENDER.currentTurn = newTurn;
4    COASTALDEFENDER.currentResources = resourceMax;
5    $("#over_map").text("Current Turn:" + COASTALDEFENDER.
         currentTurn);
6    $("#over_map_right").text("Current Resources:" +
         COASTALDEFENDER.currentResources);
7    $("#over_map_rightB").text("Current Game ID:" +
         COASTALDEFENDER.currentGameId);
8  }
```

It also runs the `removeTurnList` function that removes all turn selector objects that are of a turn greater than the one selected.

```
1  function removeTurnList(currentTurnId){
2    x = 0
3    $("#revertMeunu li").each(function(){
4      x = x + 1
5      if (x >= currentTurnId){
6        $(this).remove();
7      }
8
9    });
10 }
```

8.12.6 Summary

The necessary elements of the game have been completed. Since time allows additional functionality should be considered. At the moment the player has

no way of measuring their success, this can be introduced via critical locations. So a manner in which a user can interrogate the number of number of critical locations (perhaps via a button to change hexes colours accordingly) will be needed.

8.13 11th Cycle

8.13.1 Overview

Additional features to be included will be.

1. The inclusion of critical locations. These are hexes which have been marked critical and will be tracked throughout the game. A method to display not only how many critical locations remain but also to display visually which hexes are critical and which are not.

2. The inclusion of the ability to display the official SMP designations. So the user can adapt and change the official designations.

8.13.2 Objectives

This mission has three secondary objects and no primary objective as each component here is useful but not critical.
Apply Critical locations to the game model: So that they can be interrogated and effect game play.
Create method to display critical locations and count them: This will be done through a button that will toggle the colour of any critical location and an alert showing the current critical location count. Apply SMP designations: Create a method to display on the map the current SMP as if they were line designations by the user.

8.13.3 Creating Critical locations

Like the bias map the positioning of critical locations needs to be be determined via a mirror map that can be stored in the `COASTALDEFENDER` object as a map attribute. Like the bias map it will have the same dimensions as the `startMap` attribute. In the case of the `critLocMap` attribute a 0 counts as a non critical location whilst a 1 denotes a critical location.
MapFunctions.js

```
function removeTurnList(currentTurnId){
critLocMap : [
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,1,1,1,1,1,1,1,1,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
[0,0,0,0,0,0,0,0,0,0],
],
```

```
function NewHex(xCord,yCord,gridPoint){
...
this.isCritical = null;
...
}
```

```
function mapViewCreator(data){
...
newGrid.grid[x][y].isCritical = COASTALDEFENDER.critLocMap[x
   ][y];
..
}
```

To adjust for this in the `NewHex` objects a new `isCritical` attribute has been added. This attribute is populated during the `mapViewCreator` function. Since the hex is consider to be a critical location at all times no further amendments are required throughout the game.

8.13.4 Recording and Displaying Critical Locations

To record the number of critical locations during the `buildGrid` function after the point of creation of a `NewHex` object the function checks to see if that hex is in the same position as a critical location in the `critLocMap`. If this is the case it iterates the value of the `COASTALDEFENDER.critLocCount` by one.

```
function buildGrid(){
  ...
      if(COASTALDEFENDER.critLocMap[x][y] == 1){
```

```
4           COASTALDEFENDER.critLocCount = COASTALDEFENDER.
                critLocCount + 1;
5       }
6
7   ...
8   }
```

Displaying the critical location will be linked to a button built into the UI. Upon clicking it the JQuery trigger calls an alert to display the current number of critical locations and calls the `showCrit()` function.

mapScreen.html

```
1   $(document).ready(function(){
2     $("#displayButton").click(function(){
3       alert("Critical Location Count: " + COASTALDEFENDER.
            critLocCount);
4       showCrit();
5     });
6   });
```

MapFunctions.js

```
1   function showCrit(){
2     if(COASTALDEFENDER.showCritTrue == false){
3       for(var x=0; x<COASTALDEFENDER.g; x++){
4         for(var y=0; y<COASTALDEFENDER.i; y++){
5           if(COASTALDEFENDER.critLocMap[x][y] == 1){
6           hexShape = newGrid.grid[x][y].hex;
7           hexShape.setOptions({fillColor : COASTALDEFENDER.
                colourList.CritLoc});
8           }
9         }
10      }
11      returnValue = true;
12    }
13    if(COASTALDEFENDER.showCritTrue == true){
14      for(var x=0; x<COASTALDEFENDER.g; x++){
15        for(var y=0; y<COASTALDEFENDER.i; y++){
16          if(COASTALDEFENDER.critLocMap[x][y] == 1){
17            newGrid.grid[x][y].updateOptions();
18          }
19        }
20      }
21      returnValue = false
22    }
23    COASTALDEFENDER.showCritTrue = returnValue;
```

```
24 }
```

The `showCrit` function first checks the `COASTALDEFENDER.showCritTrue` value, if false it runs a 2D array testing each hex in the `hexGrid` to see if it is a critical location. If this is the case it calls the `setOptions` method on that hex shape to change its colour. It then sets the `showCritTrue` to true. If the initial check returns false it runs the 2D array check on all critical locations and reverts them back to the colour according to their terrain attribute.

8.13.5 Displaying the SMP designations

Like the critical locations the manner in which the SMP is designated onto the map is through a mirror map in the `COASTALDEFENDER` object.

```
1  SMPMap : [
2  [0,0,0,0,0,0,0,0,0,0],
3  [0,0,0,0,0,0,0,0,0,0],
4  [0,0,0,0,0,0,0,0,0,0],
5  [0,0,0,0,0,0,0,0,0,0],
6  [0,0,0,0,0,0,0,0,0,0],
7  [0,0,0,0,0,0,0,0,0,0],
8  [0,  htl  ,  htl  ,  htl  ,  htl  ,  htl  ,
       htl  ,  htl  ,  htl  ,0],
9  [0,0,0,0,0,0,0,0,0,0],
10 [0,0,0,0,0,0,0,0,0,0],
11 [0,0,0,0,0,0,0,0,0,0],
12 ],
```

To display the official SMP designations a JQuery trigger is linked to a DOM button element. When clicked it calls the `loadSMP()` function.

mapScreen.html

```
1  <button type="button" id="SMPButton" >Apply SMP Designations!
       </button>
```

```
1  $(document).ready(function(){
2    $( "#SMPButton" ).click(function() {
3      loadSMP();
4    });
5  });
```

MapFrunctions.js

```
1  function loadSMP(){
```

```
2     for(var x=0; x<COASTALDEFENDER.g; x++){
3       for(var y=0; y<COASTALDEFENDER.i; y++){
4         if(COASTALDEFENDER.SMPMap[x][y] == 0){
5           newGrid.grid[x][y].hasLine = "NAI";
6           newGrid.grid[x][y].hex.setOptions({strokeColor :
                COASTALDEFENDER.colourList.nullHex, strokeWeight :
                 0.5});
7         }
8         if(COASTALDEFENDER.SMPMap[x][y] == "htl"){
9
10          newGrid.grid[x][y].hasLine = "HTL";
11          newGrid.grid[x][y].hex.setOptions({strokeColor :
                COASTALDEFENDER.colourList.HTLLine, strokeWeight :
                 6});
12        }
13        if(COASTALDEFENDER.SMPMap[x][y] == "atl"){
14          newGrid.grid[x][y].hasLine = "ATL";
15          newGrid.grid[x][y].hex.setOptions({strokeColor :
                COASTALDEFENDER.colourList.ATLLine, strokeWeight :
                 6});
16        }
17        if(COASTALDEFENDER.SMPMap[x][y] == "mr"){
18          newGrid.grid[x][y].hasLine = "MR";
19          newGrid.grid[x][y].hex.setOptions({strokeColor :
                COASTALDEFENDER.colourList.MRLine, strokeWeight :
                6});
20        }
21      }
22    }
23  }
```

The `loadSMP` function runs a 2D for loop checking each `NewHex` object in the `hexGrid`. It then checks each hex against the mirror `SMPMap` and applies all values that do not equal 0 to that hex as if it were a line designation.

8.13.6 Summary

With all of the component parts for the `Coastaldefender` created all that remains is the amend the contents of the `COASTALDEFENDER` object to draw an accurate map of the North Norfolk region.

8.14 12th Cycle

8.14.1 Overview

Using the `COASTALDEFENDER` object as a controller the attributes can be manipulated to create the map that will reflect the North Norfolk coastline.

8.14.2 Objectives

This mission has a single objective.

Create the completed Map.

Augmenting the `COASTALDEFENDER` Object The final `COASTALDEFENDER` object is shown below. Each attribute that describes the shape and behaviour of the map. The first set of creator attributes `g`, `i`, `hexR`, `d` and `point` determine the size of the map, the size of the hex polygons and the position of the map. `MyOptions` is used by the instance of Google Maps to determine its setting, in this case all map controls apart from zoom are disabled.

The game attributes `currentResources` and `maxTurn` were finally used at these values after some trial and error. 10 turns in totally appears to work well in conjunction with 50 resources.

The map attributes have to match the creator attributes. These are used in turn to determine the starting map, where critical locations are the SMP designations and of course natural bias for the `seaPower` function. For the sake of brevity and formatting the contents of these maps have been removed from the example.

MapFrunctions.js

```
var COASTALDEFENDER = {
  g : 20,
  i : 60,
  hexR : 100,
  d : 2*100*Math.cos(Math.PI/6),
  point : new google.maps.LatLng(52.992367 , 0.604162),
  myOptions : {
    zoom: 13,
    center: new google.maps.LatLng(52.976245 , 0.67729),
    mapTypeControl: false,
    navigationControl: false,
    zoomControl: true,
    zoomControlOptions: {
      position: google.maps.ControlPosition.RIGHT_CENTER
    },
```

```
    panControl: false,
    streetViewControl: false,
    mapTypeId: google.maps.MapTypeId.HYBRID
  },
  JSON : null,
  currentResources : 50,
  maxResources : 50,
  critLocCount : 0,
  showCritTrue : false,
  gameStateHistory :[0],
  currentTurn : 1,
  maxTurn : 9,
  currentGameId: "",
  startMap : [
],
  biasMap : [
],

critLocMap : [
],

SMPMap : [

  modifyList : {
  //The first half of this list are modifiers according to
     the terrain type of the hex.
  'n' : 0, //null
    'm' : 0, //marine
    'b' : 10, //beach
    'd' : 12, //dune
    'sm' : 15, //saltmarsh
    'hf' : 17, //human farmland
    'hu' : 19, //human urban
  //The second half of this list are modifiers according to
     the assigned line type to the hex.
  'HTL' : 2, //hold the line
  'ATL' : 4, //advance the line
  'MR' : -2, //managed realignment
  'NAI' : 0, //no active intervention
    },

  colourList : {
  "nullHex" : "#000000",
  "marineHex" : "#0000ff",
  "beachHex" : "#ffff00",
```

```
  "duneHex" : "#ff6600",
  "saltmarshHex" : "#00ffcc",
  "humanfarmHex" : "#006600",
  "humanUrbanHex" : "#666666",
  "HTLLine" : "#33FF00",
  "ATLLine" : "#FF0000",
  "MRLine" : "#660088",
  "CritLoc" : "white",

  },
  lineCost : {
  'HTL' : 2, //hold the line
  'ATL' : 4, //advance the line
  'MR' : 1, //managed realignment
  },
}
```

8.14.3 summary

The final game appears as shown in figure 85.

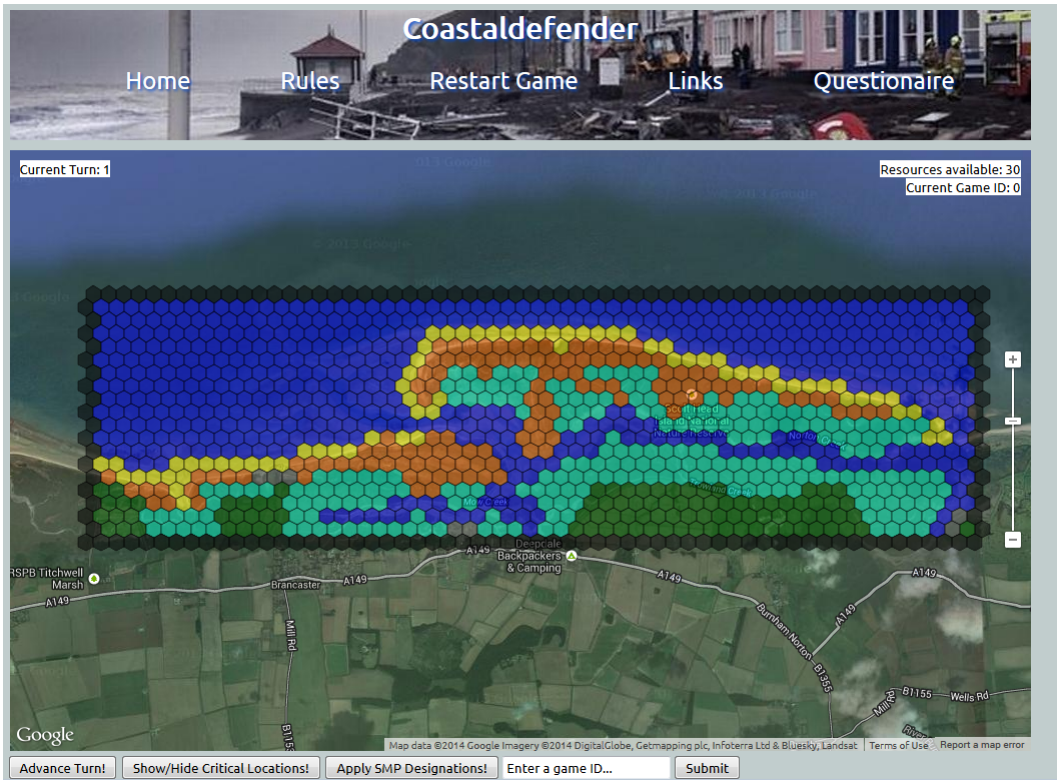

Figure 85: The game in its finished state.

References

[1] Neal Adams. Neal Adams - Science: 01 - Conspiracy: Earth is Growing!, 2007.

[2] JG Rees AJ Moores. UK Flood and Coastal Erosion Risk Management Research Strategy. 2011.

[3] Amazon. Amazon Mechanical Turk, 2013.

[4] Amazon. Amazon's Mechanical Turk. 2014.

[5] @articlePontee2010, annote = Favourite, author = Pontee, N I, file = ::, journal = Proceedings Of The Institution Of Civil Engineers, keywords = Favourite,embankments,floods,floodworks,sea defences, mendeley-tags = Favourite, number = September 2006, pages = 99–108, title

= Banked realignment : a case study from the Humber Estuary , UK, year = 2010 .

[6] Brian Ashcraft. Epic Hard FFXI Boss Killed In One Minute. *Kotaku*, 2008.

[7] Philip W. Atkinson, Steve Crooks, Allan Drewitt, Alastair Grant, Mark M. Rehfisch, John Sharpe, and Christopher J. Tyas. Managed realignment in the UK - the first 5years of colonization by birds. *Ibis*, 146:101–110, September 2004.

[8] Fiona Barclay. Hard Choices at Titchwell RSPB, 2008.

[9] Wenhong Chen Barry wellman, Jeffrey Boase. The Networked Nature of Community: Online and Offline. *IT&Society*, pages 151–165, 2002.

[10] Oliver Bennett. Flood defence spending in England. Technical report, House of Commons Libary, London, 2013.

[11] Geoffrey Boulton and James Norton. The Independent Climate Change E-mails Review Review team :. (July), 2010.

[12] I Brown. Modelling future landscape change on coastal floodplains using a rule-based GIS. *Environmental Modelling & Software*, 21(10):1479–1490, October 2006.

[13] The Goldcorp Challenge. US $ 575 , 000 Goldcorp Challenge Awards world ' s first 6 million ounce internet gold rush yields high grade results ! pages 6–8, 2001.

[14] Chinese Academy of Sciences. Beijing Declaration on Digital Earth-September 12, 2009, 2009.

[15] US Congress. UNITING AND STRENGTHENING AMERICA BY PROVIDING APPROPRIATE TOOLS REQUIRED (USA PATRIOT ACT) ACT OF 2001 An Act. pages 1–132, 2001.

[16] THE UNITED STATES DISTRICT COURT and FOR THE MIDDLE DISTRICT OF PENNSYLVANIA. KITZMILLER vs DOVER AREA SCHOOL DISTRICT, 2005.

[17] Max Craglia, Michael F Goodchild, Alessandro Annoni, Gilberto Camara, Michael Gould, Werner Kuhn, David Mark, Ian Masser, and David Maguire. Next-Generation Digital Earth. *International Journal*, 3:146–167, 2008.

[18] Destructoid. Final Fantasy XI boss causes vomiting, takes 18 hours to beat. *Destructoid*, 2008.

[19] DSDM Consortinum. *DSDM Busniess Focused Devlopment 2nd Ed.* 2nd edition, 2003.

[20] EGFI. Activity: Prevent Shore Erosion, 2013.

[21] Square Enix. Square Enix, 2013.

[22] Blizzard Entertainment. World of Warcraft, 2004.

[23] Rovio Entertainment. Angry Birds. 2013.

[24] European Commission. INSPIRE, 2007.

[25] Facebook. Facebook, 2014.

[26] Joshua Fairfield and Edward Castronova. Dragon Kill Loot Table, circa 700 CE. 1. *Hand, The*, pages 1–10, 2006.

[27] John J Fisher. Geological Society of America Bulletin Barrier Island Formation : Discussion. *October*, 1968.

[28] Barbara Forest. UNDERSTANDING THE INTELLIGENT DESIGN CREATIONIST MOVEMENT: ITS TRUE NATURE AND GOALS. (July), 2007.

[29] Django Software Foundation. The Django Project. 2014.

[30] Martin Fowler and Jim Highsmith. The Agile Manifesto. (August), 2001.

[31] Peter Frew. An Introduction to the North Norfolk Coastal Environment. *Coastal Management*, pages 1–25, 2009.

[32] Fraxis Games. Civilization Series, 2013.

[33] GDSI. From The SDI Cookbook Welcome to the SDI Cookbook From The SDI Cookbook. (November), 2008.

[34] Clickworker GMBH. Clickworker. 2014.

[35] Google. Google Earth Image, 2013.

[36] Google. Google Maps API, 2014.

[37] Google. Google Maps Help, 2014.

[38] Al Gore. The Digital Earth : Understanding our planet in the 21st Century. *Earth*, 1998.

[39] Greater London Authority. London Data Store, 2010.

[40] Dean Hall. DayZ, 2011.

[41] Simon K Haslett. *Costal systems*. Routledge, London, 2000.

[42] Scottish Natural Heritage. Scottish Natural Heritage: A guide to managing coastal erosion in beach/dune systems, 2013.

[43] John H Hoyt. Geological Society of America Bulletin Barrier Island Formation : Reply. *October*, 1968.

[44] Goldcorp Inc. Goldcorp. 2013.

[45] Discovery Institute. wedge document. Technical report, 1999.

[46] Bohemia Interactive. Bohemia Interactive, 2013.

[47] Panos Ipeirotis. Why Mechanical Turk Allows Only US-based Requesters? 2010.

[48] Rebecca Willis James Wilsdon. See Through Science: Why public engagement needs to move upstream. Technical report, DEMOS, London, 2004.

[49] Adam Jones. *Genocide: A Comprehensive Introduction*. Routledge, New York, 2nd edition, 2010.

[50] Rob E De Jong, Mark H Lindo, Saeed A Saeed, and Jan Vrijhof. Execution Methodology for Reclamation Works Palm Island 1. *Terra et Aqua*, (92), 2003.

[51] Ted Kaczynski. The Unibomer Manifesto, 1995.

[52] Rachel Kaufman. "GREEN PEA" PICTURES: New Galaxy Class Discovered, 2010.

[53] Paul Kelso. Greenpeace wins key GM case, September 2000.

[54] Percival Davis Kenyon and Dean H. *Of Pandas And People*. Foundation for Thought and Ethics, 1st edition, 1989.

[55] Henrik Kniberg. Scrum and XP from the Trenches. pages 1–90, 2006.

[56] Linux.org. Linux.org, 2013.

[57] Life Magazine. *100 pictures that changed the world*. 2003.

[58] Jane McGonigal. *Reality is Broken*. Random House, St Ives, 1 edition, 2011.

[59] Competitive Advantage via Quantitative Methods. The Beginning of Crowdsourced Analytics, 2012.

[60] Demers Michael. *Fundamentals of Geographic Information Systems*. John Wiley & Sons, inc, Hobokem, 4th edition, 2009.

[61] Microsoft. Bing Maps, 2013.

[62] Christopher Moore. Hats of Affect: A Study of Affect, Achievements and Hats in Team Fortress 2. *Game Studies*, 11(1), 2011.

[63] NASA. Earthrise, 1968.

[64] NASA. Landsat Science, 2013.

[65] NASA. World Wind, 2013.

[66] NASA. NASA's projection series, 2014.

[67] Gabe Newell. Gabe Newell: Reflections of a Video Game Maker, 2013.

[68] SCOPAC (Standing Conference on Problems Associated with the Coastline). Shoreline Management Plans, 2013.

[69] open street Maps. Open Street Map. 2013.

[70] PBWORKS. blackpoolsixthasgeography, 2013.

[71] N I Pontee. Banked realignment : a case study from the Humber Estuary , UK. *Proceedings Of The Institution Of Civil Engineers*, (September 2006):99–108, 2010.

[72] Jenny Preece, Diane Maloney-krichmar, and Chadia Abras. History of online communities. pages 1023–1027, 2003.

[73] Reality Prime. Notes on the origin of Google Earth, 2006.

[74] Raptr. Community as a Service, a New Paradigm. Technical report, 2012.

[75] Reddit. Reddit, 2014.

[76] Reddit. Subreddit Drama. 2014.

[77] T. Rogers-Hayden and N. Pidgeon. Moving engagement "upstream"? Nanotechnologies and the Royal Society and Royal Academy of Engineering's inquiry. *Public Understanding of Science*, 16(3):345–364, July 2007.

[78] Julie Dean Rosati, Robert G. Dean, and Gregory W. Stone. A cross-shore model of barrier island migration over a compressible substrate. *Marine Geology*, 271(1-2):1–16, May 2010.

[79] Jean-Jacques Rousseau. *Discourse on Inequality*. Paris, 1754.

[80] RSPB. Titchwell Marsh RSPB Reserve : EIA Scoping Report. Technical Report January, 2008.

[81] RSPB. Titchwell Marsh RSPB Reserve Coast Protection Works : Non Technical Summary. Technical Report November, 2008.

[82] Bertrand Russell. *The History of Western Philosphy*. Routeledge, London, 2nd edition, 1946.

[83] Dan Simmons. *Ilium*. Harper Collins, 2003.

[84] Skyscrapercity.com. Sky Scraper City Forum, 2008.

[85] The British Cartographic Society Soc. How Long is the UK Coastline?, 2013.

[86] British Geographic Society. Coastal Erosion. Technical report, British Geographic Society, 2012.

[87] The Royal Society. The Public Understanding of Science. Technical report, The Royal Society, London, 1985.

[88] 'SRI International'. Digital Earth, 2001.

[89] John A Steers. *Scolt Head Island*. 1934.

[90] David R. Stoddart, Denise J. Reed, and Jonathan R. French. Understanding Salt-Marsh Accretion, Scolt Head Island, Norfolk, England. *Estuaries*, 12(4):228, December 1989.

[91] Joyce Tait. Upstream engagement and the The shadow of the genetically modified crops experience in Europe. *Embo Reports, European Molecular Biology Organization*, 10(Special Issue), 2009.

[92] Simon Tatham. PuTTY, 2014.

[93] Frank Taylor. Google Earth Blog, 2014.

[94] Android Development Team. Compatibility Program Overview, 2013.

[95] Unity Technologies. Unity 4, 2014.

[96] The Environment Agency. North Norfolk Shoreline Management Plan Draft Plan. Technical report, 2010.

[97] The Environment Agency. North Norfolk Shoreline Management Plan Final plan. (November), 2010.

[98] The Environment Agency. National flood and coastal erosion risk management strategy for England. Technical report, The Enviroment Agency, Defra, Bristol, 2011.

[99] The National Archives. The open goverment Licence, 2010.

[100] Linux Torvalds. LINUX's History. 1992.

[101] R.K. Turner, D. Burgess, D. Hadley, E. Coombes, and N. Jackson. A costbenefit appraisal of coastal managed realignment policy. *Global Environmental Change*, 17(3-4):397–407, August 2007.

[102] Twitter. Twitter, 2014.

[103] UW Department of Computer Science UW Center for Game Science and Engineering. foldit, 2013.

[104] Vinod Valloppillil. Halloween Document, 1998.

[105] Webfaction. Webfaction, 2014.

[106] Josie Wernecke. *The KML Handbook*. Addison-Wesley Professional, Reading, Massachusetts, 2008.

[107] Don Topscott Williams and Anthony. *Wikinomics*. Penguin Group Publishing, 2010.

[108] Galaxy Zoo. Galaxy Zoo, 2013.