Method selection and planning:

Team 23

Tom Rosewell

Sam Redgewell

Tom Keeler

Jake Stogden

John Limb

Adham Noureldin

Part A: Software Engineering Methods

Our team decided to use the scrum software engineering method **[1]**, because not only is our project short-term, but flexibility throughout its development is paramount. This lends itself towards the scrum method because there will be frequent incremental progress and change that necessitate ongoing communication and transparency. This was chosen in favour of the more plan-oriented approaches seen with the Capability Maturity Model **[2]** and the Team Software Process **[3]**. We concluded that these approaches were neither flexible nor agile enough and were better suited towards a longer-term project unlike ours.

With this, we decided to delegate scrum tasks to be carried out over a two week period, and convene a virtual meeting once a week, in order to ensure that sufficient progress was being made, or to delegate new tasks for the next scrum - depending on whether the meeting was taking place halfway through a scrum, or at the end.

For the second phase of the project, our team decided to continue using the scrum format for the previously listed reasons, albeit with one caveat, each sprint would be longer than typical (double the length of a standard sprint in this format). This is to counteract any complications that may arise, thus limiting their effect on the timeframe of this project.

Our team used three tools for facilitating the completion of tasks:

- **Discord** - team communication and task delegation
- **GitHub** - collaborative programming
- **Google Drive** - team access to and writing of documentation

<u>Discord:</u>

From 15th November, our team used each others' email addresses to communicate. Following this, however, at the first in-person practical on 20th November, we each agreed to use Discord as the main tool of communication instead. This decision was arrived at due to the many useful features that it offers in contrast to, say, Snapchat or Whatsapp.

- A separate channel can be created for each independent aspect of the project, therefore facilitating organised, and aggregating relevant, communication. **See appendix 1.**
- All team members were already familiar with Discord at the time of making the decision, hence time was not wasted in becoming acquainted with a new communication tool.
- Notifications from team members are made apparent, thus reducing the likelihood of team members missing important updates, messages or questions.
- Discord allows a user to seamlessly copy and paste screenshots, links and upload files, therefore facilitating accurate communication.
- Task delegation can be carried out with ease, because messages can be 'pinned', making them visible to all users at all times.

<u>GitHub:</u>

There was not much debate over which tool to use for collaborative programming - GitHub is the most popular git repository today, with the largest number of users and projects. **[4]**

- GitHub allows for multiple people to work on different parts of a programming project simultaneously, on their local machine, whilst maintaining a global repository that is updated through push/pull changes. Hence, separate programming tasks can be easily assigned to individual team members; and should an individuals' work get lost, the whole project will not be lost.
- All team members were already familiar with GitHub at the time of making the decision, hence time was not wasted in becoming acquainted with a new collaborative programming tool.
- All pull/merge requests can be reviewed and all changes between versions of the project are tracked through GitHub. Therefore, users' input can be accurately tracked, and any programming mistakes can be easily identified and rectified.

<u>Google Drive:</u>

It seemed rather obvious to write the documentation through a collaborative platform, like Google Drive, instead of something like Microsoft Word. Although our team initially considered OneDrive, we ultimately decided to use Google Drive for the writing of documentation, because we each already had an email address associated with Google, and so wouldn't have to spend much time getting things set up.

- We decided to use a collaborative application, because it allows for small changes and refinements to be made to a document, by anyone, without the need for the 'owner' of the document to send it to them.
- All team members were already familiar with Google Drive at the time of making the decision, hence time was not wasted in becoming acquainted with a new collaborative documentation tool.
- Google Drive allows you to track the various changes to a document, as well as who made them. So it allows you to gauge the input of individual team mates, as well as restore an old copy of the document, if necessary.

Part B: Approach to Team Organisation

Allocation of Roles - Internal Organisation:

Given that we had a small team of five members, with only three broad categories of task (website, java implementation and documentation) we thought it would be superfluous to assign numerous elaborate roles to one another, such as 'Java Leader' and 'Website Leader' etc., as can so often be the tendency with projects of this kind. (This is not to say that we did not assign an overall project leader.) So instead, we simply took into account each individuals' experience and preferences and then assigned tasks accordingly: people were assigned to roles for which they either had experience or expressed an affinity towards.

- Josh Q was assigned overall project leader due to his clear enthusiasm and communicability;
- Louis H and Lewis P were assigned to the website from the get go due to their past experience in this area;
- Faris A and Josh Q were assigned to the Java research, and later the Java implementation, due to their experience and aptitude for programming 2D games;
- and Kyle M was assigned to the documentation due to his predilection for writing and expressed inclination to take note of details.

➢ Once the website was soon complete, after sprint two (elaborated in part C), Louis H helped with Java and Lewis P helped with documentation.

Once these roles were established, our group decided to utilise the division of labour, and essentially leave people working within the area to which they'd be assigned, thereby cultivating an environment wherein people are familiar with the work that they are undertaking.

This is not to say, however, that members were at any point oblivious to the ongoings of another members' work: our in-person and online meetings were utilised for this exact purpose - to ensure that every member was on the same page at all times, and that, if at any point necessary, an individual could move from one disparate aspect of the project to another in a relatively seamless fashion.

For the continuation of the development, we had a team of six members, now with two main tasks: creating/updating documentation, and java development. Due to a range of talents and areas of expertise, the group was divided up bearing these in mind.

- Jake and Tom K were assigned to updating documents
- Adham and Tom R were assigned to implementation of new code
- John and Sam were assigned to testing and updating requirements as new bugs were found
- Tom K was also assigned to managing and updating the website

Much like our predecessors, these roles weren't set in stone; our team members worked in a fluid fashion and would help in other aspects of the project as and when needed.

External Organisation:

In accordance with the principles outlined by the SCRUM method, we decided to meet virtually on a Wednesday afternoon each week, through Discord.

If assigned work could not be completed within the allocated time frame, concerns about this were preemptively raised through the relevant Discord channel, and additional teammates would be assigned as a consequence. In this, they would abandon a task of theirs that was of comparatively less important, help the teammate, and then return to their task as soon as possible afterwards.

Through the second half of the project we initially continued to use a SCRUM method with extended sprints, however found this to be a poor fit for our group, hence our change to a plan driven method around the later half of the Easter holidays.

Part C: Systematic Plan

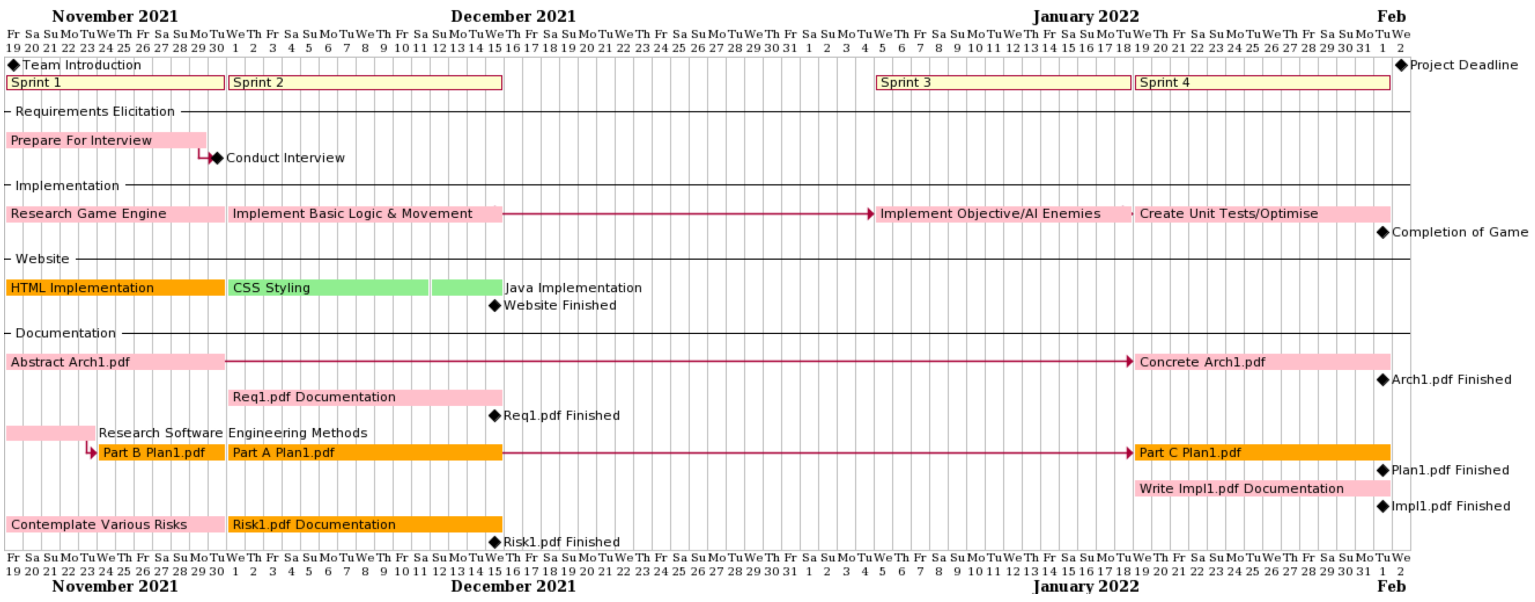
On the Wednesday of each week, our group would meet and if it was the end of a scrum sprint, we would plan out the following week using the Discord 'Task Delegation' channel; if it was during the middle, we would ensure that everything is going to plan. Each task delegated amongst our group was assigned a task priority of either low, medium or high - assigning a priority would help not only with motivation towards completing the task, but also with the aforementioned scenario of deciding which task to neglect in order to help a teammate who's fallen behind on something.

Task priority was decided upon

- How many marks the proper completion of the task would accrue
- The perceived size of the task; the number of necessary people it involved
- The inherent difficulty of the task
- The number of other contemporary tasks (and their priorities)

The decision of which tasks were to be assigned mainly depended upon how many tasks during the previous week/sprint had been finished. If some of them were unfinished, they were to be reassigned for the next week, their priority potentially moved up, and an additional individual may have been subsequently assigned to it.

With this, an initial Gantt chart was designed using PlantUML **[5]** in order to illustrate the various task dependencies, their respective priorities and whether we were on schedule or not. For this purpose, green was used to represent low-priority tasks, yellow for medium-, and red for high-. The Gantt Chart was also split into separate sections for each of the deliverables to allow for better readability. This was posted on Discord so that it could be seen by team members at all times, thus ensuring everyone was always on the same page. (If image resolution is too small, a link to it can be found to it at **[6]**)



Our overall project plan, illustrated by the Gantt Chart, was delineated whilst taking into account

- The number of team members,
- The amount of requisite work,
- How long we had,
- What dependencies between tasks there were,
- How long those tasks would take.

With these in mind, at the highest level of abstraction, we conceptualised our project as consisting of three main work packages, which then ramified into five milestones, and from here, with our deliverables in mind, we were able to specify individual tasks. These tasks were defined numerically according to which deliverable they were associated with e.g. T6.**X** is a task corresponding to the implementation of the game, because the game implementation is D**6**.

Work Packages

- WP1 ⟶ Deliverables
- WP2 ⟶ Website
- WP3 ⟶ Programming

Milestones

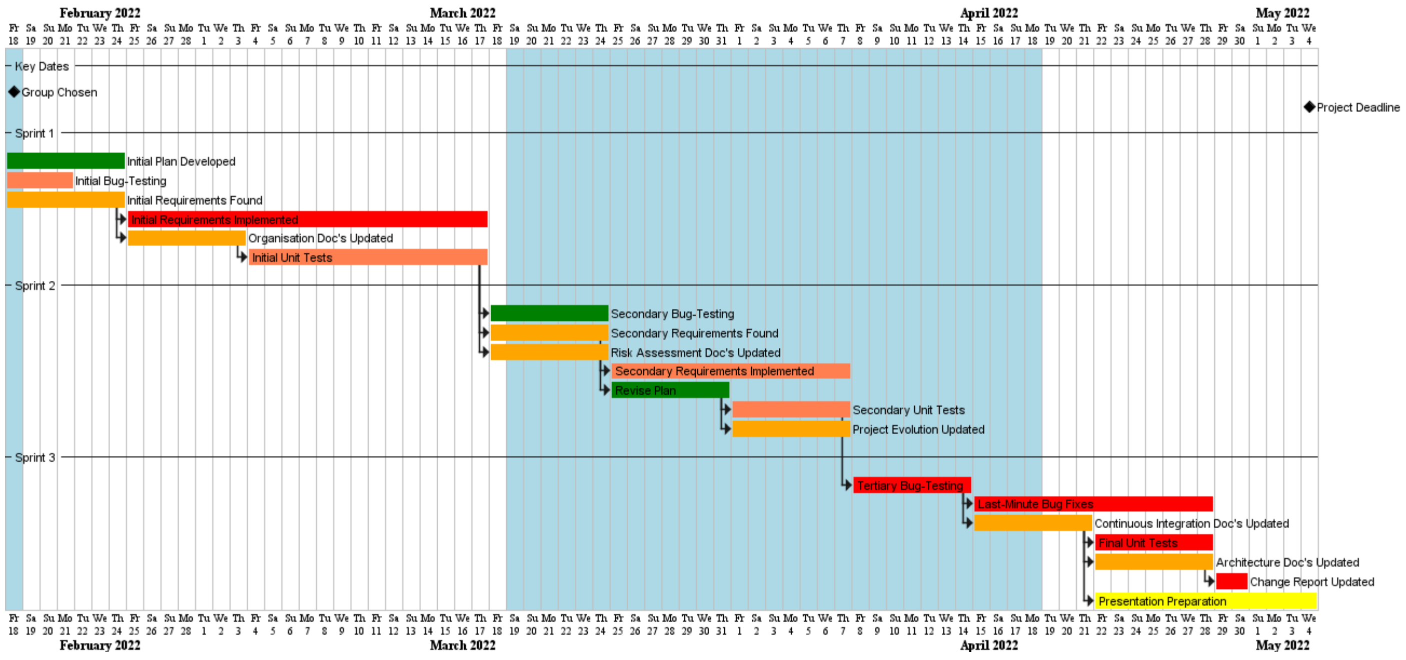
We used milestones to keep track of important achievements of the project. These were outlined as follows

M1 ⟶ Requirements Elicitation	Due 30/11/2021
M2 ⟶ Website Creation	Due 30/11/2021
M3 ⟶ Initial Skeleton of The Game Programmed	Due 10/12/2021
M4 ⟶ Completion of Game and Corresponding Tests	Due 01/02/2022
M5 ⟶ Completion of All Deliverables	Due 01/02/2022

Deliverables

D1 ⟶ Requirements	Due 15/12/2021
D2 ⟶ Website	Due 19/12/2021
D3 ⟶ Risk Assessment and Mitigation	Due 15/12/2021
D4 ⟶ Method Selection and Planning	Due 01/02/2022
D5 ⟶ Architecture	Due 01/02/2022
D6 ⟶ Implementation of Game	Due 01/02/2022

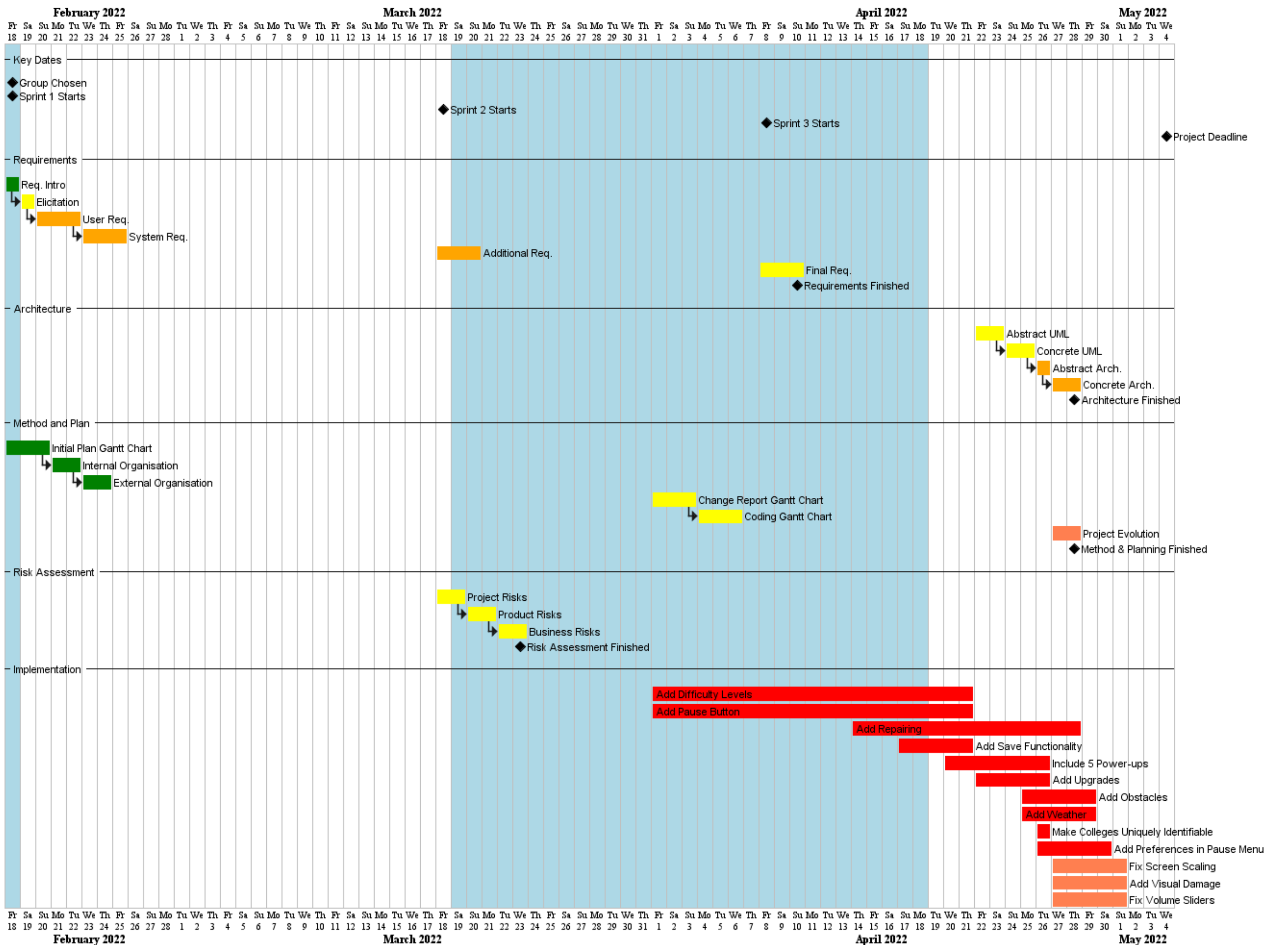
Through the first week of taking over this project, our team developed a plan utilising the principles of SCRUM development in order to maintain consistency. We broke the task into three sprints of (on average) four weeks long, in which we would have a cycle of bug testing, fixing said bugs and implementing new requirements, and testing. These would be organised through weekly discord/in person meetings, where any queries could be addressed and resolved. At the end of each sprint, the website would be updated to reflect the new changes and outline goals for the next sprint.



The Gantt chart above, displays our original project plan, which was designed using similar principles as to the previous group, in an effort to provide consistency. Colours were limited in use to aid in clarity for the reader, with a colour scale of red to green, where red means urgent/extremely important and green reflects less important tasks, and blue was chosen to display the Easter break, a period where the plan may stray further askew than expected.

This plan was created with delays in mind, be that through missing team members or otherwise, in order to give us the best chance at maintaining our schedule. The decision was made to implement multiple sprints, to enable our team with greater flexibility, thus if certain aspects weren't fully completed within their original allotted time frame, they could be rolled over into the next sprint.

Due to extra unforeseen delays earlier in the development process, our team adopted a solely plan driven approach. This is displayed in the Gantt chart below:



Although not included in the Gantt chart, we planned to have unit testing for each new feature immediately after it was implemented.

<u>Project Evolution:</u>

Due to our team's general pragmatism and prudence whilst developing the overall project Gantt Chart, there weren't many unanticipated obstacles that emerged, causing us to diverge from our plan.

With that being said, during sprint 1 of the project, we had elected to have two weekly meetings (excluding the in-person practical on a Friday), but soon realised that this was counterproductive, because not enough time had elapsed for anything meaningful to have changed since the last meeting, and there was therefore nothing new to discuss that simply couldn't have been typed in the relevant Discord channel instead. Therefore, going forward, with sprint 2, 3 and 4, we met (online) just once a week on Wednesday, which worked much better for our group.

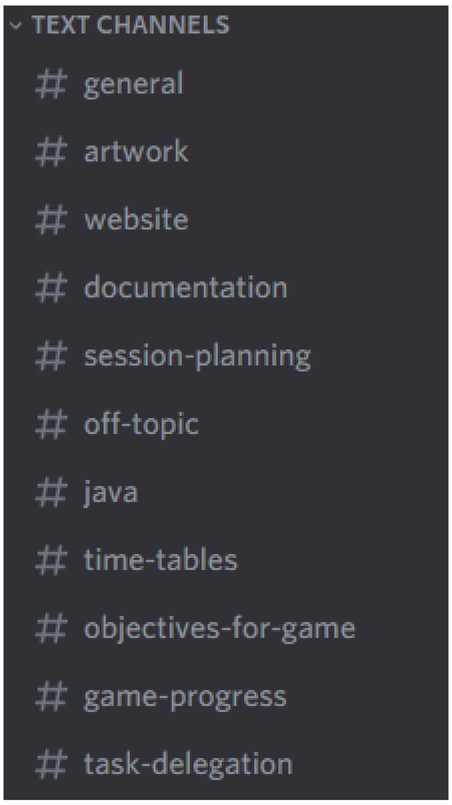
More specifically, during the fourth sprint, after our exams, it became apparent that there was more work to do than we had originally envisaged; specifically in getting the boat to move normally across the map (T6.4). To account for this, during our meeting on 19/01/2021 (sprint 4), our group made the decision to move Louis H from helping Kyle M with part C of the Plan1.pdf documentation (T4.3), towards assisting with this programming issue. This decision was based primarily upon the fact that T4.3 was only a medium-priority task, as per the Gantt chart, and T6.4 was a high-priority task.

The second half of the assessment started off following the plan, with a few delays to the implementation of new code, but as briefly mentioned in the systematic plan section, by the Easter holidays we were significantly behind schedule with regard to the implementation side of our original plan, however remained consistent with regard to the updating of documentation.

Due to this, we devised a new plan to approach the task of implementing new requirements and testing our code. This new plan was followed closely, with team members being moved from documentation to implementation and testing in order to meet these strict deadlines. These new deadlines were consistently met, allowing the testing of said features to be started by the testing group, while the implementing team moved on to other features. As features were tested, any bugs from these tests were reported back to the implementing team, allowing bugs to be quickly patched.

__Appendix:__

1.



Individual text channels for the various aspects of the project

Bibliography:

[1]
https://www.digite.com/agile/scrum-methodology/#:~:text=Scrum%20is%20an%20agile%20development,an%20iterative%20and%20incremental%20processes.&text=The%20primary%20objective%20of%20Scrum,collective%20responsibility%20and%20continuous%20progress.

[2] https://www.michigan.gov/documents/CMM_39213_7.pdf

[3] https://resources.sei.cmu.edu/asset_files/TechnicalReport/2000_005_001_13754.pdf

[4] https://usersnap.com/blog/gitlab-github/

[5] https://plantuml.com/

[6]
http://www.plantuml.com/plantuml/png/fLVDKjiy5DtxAUxg-rPfuW33TDCXuQUb0mD3qhB8i53iYwFYIAwa9CEkhz7NwvDKTa0ZXolaJ6SyBE9pZixzkLVcH1kcJC64CScjaZymCz3yf64u6ARzr3uVazaKsH8kXL4oNsM6I_402rPLA3GmqqA-YfmXoqikS1Sux0_gXmBcN5SbUzOW1MYsH51o0qm3SNeocrIA2mFfmwjTAJmVm4GE9TD6Gpg4t8ewnl0jmnfAVSPHct6mmwZX_S6HppXiCmufXdFbCpnRuVXTh9wbCDe9UdR2Q2UO9BqUtE7F5LUuH64qd9SyuuRL2OTUhwuOLamXN4XLLmFLckCcdlzq6wbPe83PgHHr3Nq-qKW70viOwoNzKEMFS2QpLUseSTAzUev_mGNN2ynRHuNa_zU08Wy6opLaifGAS-02AYwUmaO9R9tZosLLeeCrcRr3ZKnb2_XSb-PS55nWzyPsWZ1cccTm9Glxzp-ubklcvqufSoNov6xczOJoDVP7bzOLBZdgZhFWwgsG6OHlWXkOedtVlwaCNtAD3pjusnhrmKrEv6EJ2RzWxqhz-VLxMxPuCkWIqWKWaD7iK9omSneVZo4qpdGWCTVxwuKZsbkh7ebxd6jki9c5BzFhgvqPwJuAfvC9JCnpoKKHx_hqXVILhLdqnA5zFB9RtwJTTb6bOgB07NDjICcBXIaKeWZO2SDVSaTlEKBChJYR-FR6Qj8_cckZM6PWP2lLhBNEsNSRFbDeC2WGlWndZa97GsYxOAXtmLQaegH39yRJdHJ1jGiUz49s4Dq5OGt_fCHRPnFvQ3PClKuRAWk0QpGBcUkc05CfInsjSEj-kBLG6CDjoSGs-1VGyHQqVJ_otiUl7vzvIZ2ZZUCeWSRnbTjjSw-umIRHn1tUrG1iD-mx3YBd-5riq8AO1NzdYikLXZkkdtJt7LZ3_-t3oGdGeOGEY0T1pnEj5rgZONnuFCA6zXuMCWQYwMdpVYF6SUCOBjA-NQtsFX4JGW632GwnnztP8v-T9Ax7Zy78LiCgV7CLpiiL9Xx6OwH7kundB4kvIRoFLoFzwGuUu64SWxRaRkVa14NU_7Vz5m00