**RISK ANALYSIS**

**Game xếp hình for ZP**

**1-Giới thiệu**

***1.1-Mục đích tài liệu***

Xác định các rủi ro có thể xảy ra trong dự án. Đánh giá mức độ nghiêm trọng và khả năng xảy ra của từng rủi ro. Đề xuất giải pháp hạn chế và đối phó với rủi ro.

***1.2-Phạm vi tài liệu***

Tài liệu sử dụng cho game xếp hình for ZP.

***1.3-Lịch sử phiên bản***

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên phiên bản** | **Ngày thực hiện** | **Người thực hiện** | **Ghi chú** |
| 1.0 | 06/01/2016 | Nguyễn Tuấn Trí | Bản đầu tiên |

***1.4-Tổng quan tài liệu***

Tài liệu gồm 2 phần

Phần 2.1 sẽ đưa ra tổng quan về các rủi ro trong dự án. Bao gồm: tên rủi ro, mô tả, khả năng xảy ra, mức độ nghiêm trọng và độ quan trọng.

* Tên rủi ro: Một thuật ngữ hoặc một cụm từ hoặc một mệnh đề mô tả ngắn gọn về rủi ro.
* Mô tả: Giải thích chi tiết cụ thể hơn về rủi ro để giúp hình dung rõ hơn nếu cần thiết. Có thể kèm theo ví dụ cụ thể, nguyên nhân, ảnh hướng của rủi ro, ...
* Khả năng xảy ra: Đánh giá khả năng xảy ra của rủi ro với thang điểm từ 1 đến 5. (note: cần có giải thích cụ thể cho từng mức)
* Mức độ nghiêm trọng của rủi ro: Đánh giá mức độ nghiêm trọng nếu rủi ro xảy ra với thang điểm từ 1 đến 5. (cần có giải thích cụ thể cho từng mức)
* Độ quan trọng: Được tính bằng tích của khẳ năng xảy ra với mức độ nghiêm trọng của rủi ro. Những rủi ro có độ quan trọng cao sẽ được xem xét trước và cần chú ý nhiều hơn.

Phần 2.2 sẽ đưa ra các giải pháp cụ thể cho từng rủi ro theo thứ tự ưu tiên (độ quan trọng).

**2-Phân tích rủi to**

***2.1-Tổng quan về các rủi ro***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Tên rủi ro** | **Mô tả** | **Khả năng xảy ra** | **Mức độ nghiêm trọng** | **Độ quan trọng** |
| 1 | Phạm vi dự án thay đổi | Chẳng hạn: game design thay đổi, bổ sung thêm item mới, kích thước bảng chơi đa dạng hơn (không chỉ 7\*7 nữa), có thêm âm thanh nhạc nền, thêm cài đặt, kết nối với mạng xã hội, thêm payment, cần phát triển server side, không chỉ tích hợp vào zingplay mà còn phải build trên nền tảng android, ios... | 1 | 4 | 4 |
| 2 | Các thành viên gặp phải các vấn đề về công nghệ | Thiếu kiến thức và kinh nghiệm lập trình với cocos nên việc code một tính năng nào đó có thể rất lâu. Rồi khi gặp bug, để fix được nó cũng là cả một vấn đề nan giải. | 5 | 2 | 10 |
| 3 | Thiết kế cẩu thả | Vì thiếu kinh nghiệm nên nhìn nhận vấn đề không toàn diện, không đầy đủ tất cả các khía cạnh. Coi thường thiết kế, thiết kế cẩu thả, dẫn đến bản thiết kế hoàn toàn vô giá trị, hoặc lúc thiết kế chỉ nghĩ đến logic, không quan tâm đến feeling, đến khi lập trình phát hiện ra thì đã quá muộn... | 4 | 5 | 20 |
| 4 | Game thiếu feeling | Game có thể hoàn thành theo đúng logic mô tả trong game design nhưng về mặt feeling thì hầu như không có. Chẳng hạn, không thể hiện được hiệu ứng các item rơi sao cho chân thật và sống động nhất... | 4 | 4 | 16 |
| 5 | Sự hợp tác, giao tiếp giữa các thành viên kém hiệu quả | Không truyền đạt thông điệp hiệu quả cho người kia. Thông điệp ở đây được hiểu là ý tưởng, bản thiết kế... Xảy ra việc hiểu sai, hiểu không trọn vẹn.  Xảy ra các mâu thuẫn, ví dụ: cả hai thành viên mỗi người đưa ra một giải pháp khác nhau cho cùng một vấn đề và đều cho rằng giải pháp của mình đúng, phản đối giải pháp của người kia 🡪 tranh cãi 🡪 mâu thuẫn. | 4 | 4 | 16 |
| 6 | Chậm deadline |  | 2 | 5 | 10 |

***2.2-Giải pháp cho từng rủi ro***

**2.2.1-Thiết kế cẩu thả (20)**

* Đảm bảo rằng bản thiết kế là kết quả của một quá trình làm việc nghiêm túc chứ không phải chỉ làm nhanh cho xong và làm xong không dùng đến. Bằng cách:
* Tuân thủ đúng quy trình, thiết kế kiến trúc tổng quan, đưa ra các phân tích đánh giá để khẳng định rằng đó là một kiến trúc tốt ). Sau đó mới đi vào thiết kế chi tiết, các lớp có thuộc tính gì, phương thức gì, thuật toán cụ thể của từng phương thức.
* Khi thiết kế, liên tục tiến hành review, một thành viên trình bày thiết kế của mình, thành viên kia lắng nghe để hiểu và yêu cầu giải thích các điểm chưa rõ ràng hoặc chưa hợp lý.
* Nếu có thời gian, nên bổ sung thêm các kiến thức về thiết kế.

🡪Các giải pháp trên để hạn chế rủi ro, đây là một rủi ro mà nếu xảy ra thì sẽ có hậu quả rất nghiêm trọng nên bằng mọi cách không để cho nó xảy ra. Nếu xảy ra, rất khó tìm được giải pháp khắc phục thực sự hiệu quả, sẽ bị sa lầy trong những thứ do chính mình tạo ra.

**2.2.2-Game thiếu feeling (16)**

Cần đưa việc đảm bảo feeling của game thành một yếu tố quan trọng trong quality management. Cả QA và QC đều cần chú ý đến việc đảm bảo feeling nhưng trọng tâm sẽ đặt vào QA. Có nghĩa là feeling cần được quan tâm và test trong suốt quá trình phát triển game. Một số task gợi ý để đảm bảo game có được một feeling tốt:

* Nghiên cứu feeling của các game tương tự, ví dụ: candy crush saga, candy crush soda saga, ... Đưa ra bảng thống kê feeling của các game này, phân tích, so sánh để hiểu được feeling của họ làm như thế nào, có gì hay, có gì hấp dẫn. Nên quay video lại và cắt ra các đoạn quan trọng để làm tư liệu nghiên cứu sau này.
* Xây dựng đặc tả yêu cầu cho feeling. Ví dụ:

Khi người chơi ăn một cụm icon thì các icon mới được sinh ra phải có hiệu ứng rơi sao cho chân thật và sống động nhất. 3 yêu cầu sau phải được thỏa mãn:

* Các icon mới sinh ra không rơi xuống đồng thời cùng một lúc mà rơi xuống cách nhau một khoảng thời gian ngắn nhất định 🡪 tạo cảm giác là có ai đó đang thả icon xuống.
* Các icon rơi có gia tốc 🡪 tạo cảm giác tồn tại trọng trường.
* Khi đã rơi đến vị trí đích thì có hiệu ứng nảy 🡪 tạo cảm giác đàn hồi. (Lưu ý: biên độ nảy phải nhỏ để đảm bảo icon không nảy ra ngoài phạm vi ô khác và thời gian nảy phải nhanh, không làm mất thời gian của người chơi).
* Chuyển yêu cầu feeling vào thiết kế. Có nghĩa là khi thiết kế không chỉ quan tâm đến yêu cầu về mặt logic mà còn cần quan tâm đến cả yêu cầu về mặt feeling. Một gợi ý cho thiết kế là: tách biệt phần logic và phần feeling (bằng cách sử dụng mvc design pattern) để có thể dễ dàng thay đổi feeling khi cần thiết, đảm bảo khi thay đổi feeling thì không động chạm nhiều (hoặc hoàn toàn không động chạm gì) đến logic.
* Test feeling liên tục sau khi hoàn thành mỗi component. Khi test, không chỉ quan tâm đến việc component đó chạy đúng logic mà còn cần đảm bảo component đó có đủ các feeling như đặc tả. Nếu feeling đã đúng đặc tả nhưng cảm nhận vẫn chưa thực sự ổn thì cần đặc tả lại cho feeling đó và fix lại.

**2.2.3-Sự hợp tác, giao tiếp giữa các thành viên kém hiệu quả (16)**

* Để việc truyền đạt thông điệp được hiệu quả:
* Với người nói
* Không trình bày ý tưởng của mình khi đó là một ý tưởng không rõ ràng. Mình phải hiểu nó thật rõ ràng trước khi có thể trình bày cho người khác hiểu.
* Hãy hỏi lại sau khi trình bày để đảm bảo người nghe đã hiểu đúng ý tưởng của mình.
* Với người nghe:
* Lắng nghe chủ động.
* Nếu có bất cứ điểm nào chưa rõ, hoặc thấy không hợp lý, có thể yêu cầu người nói giải thích thêm.
* Để các mâu thuẫn được giải quyết hiệu quả
* Cần hiểu nhau, mỗi người đều cần hiểu thật kỹ giải pháp, ý tưởng, quan điểm của người kia trước khi khẳng định hay phủ định nó. Nếu chưa hiểu rõ, hãy yêu cầu đối phương giải thích.
* Khi xem xét, đánh giá các giải pháp cần khách quan.
* Nhờ sự tư vấn của các mentor nếu cần thiết.

**2.2.4-Chậm deadline (10)**

* Lập kế hoạch cụ thể về công việc, đưa ra các mốc thời gian cho các milestone chính. Milestone được hiểu là các cột mốc quan trọng trong dự án. Việc đưa ra các milestone có thể linh động tùy vào đặc điểm riêng của dự án không nhất thiết phải theo một chuẩn chung để tự làm khó mình. Tuy nhiên cũng nên cân nhắc để đưa ra các milestone sao cho hợp lý.
* Kế hoạch nên hoàn thành sớm ít nhất là 5 ngày trước deadline, thời gian còn lại là thời gian dự trữ.
* Nếu chậm deadline cho một milestone nào đó, hãy xem xét nguyên nhân và đưa ra giải pháp khắc phục.
* Tuy nhiên, luôn **ưu tiên chất lượng**. Không vì chậm thời gian mà mà hy sinh chất lượng. Chất lượng được hiểu không chỉ là chất lượng của sản phẩm cuối cùng mà còn là chất lượng trong từng công việc trong dự án, chất lượng của mỗi artifact được tạo ra.

Một số ví dụ cụ thể cho thấy dự án có chất lượng kém:

* Bản đặc tả feeling viết cẩu thả, thiếu sót rất nhiều feeling quan trọng, đặc tả cho từng feeling thì sử dụng ngôn từ không rõ ràng, mập mờ, khó hiểu.
* Việc test cẩu thả dẫn đến không phát hiện sớm lỗi để rồi sẽ phải trả giá sau này.

**2.2.5-Các thành viên gặp phải vấn đề về công nghệ (10)**

* Có một giai đoạn làm quen với công nghệ.
* Nguồn tham khảo API của cocos:

Project test: Đây là nguồn tham khảo API hiệu quả. Nhất là với các newbie (người mới) thì nên tham khảo ở đây trước khi đọc API Reference trên trang chính thức của cocos.

* Nếu vẫn gặp vấn đề thì xin sự trợ giúp của các mentor.

**2.2.6-Phạm vi dự án thay đổi (8)**

Giảm nhẹ các ảnh hưởng mà nó gây ra bằng cách đầu tư thời gian thiết kế một kiến trúc tốt. Hoặc có thể lường trước các thay đổi để cân nhắc trong thiết kế.

Nếu thực sự xảy ra chuyện thay đổi phạm vi dự án thì nên bình tĩnh nhìn nhận, phân tích thấu đáo vấn đề, đưa ra các ý tưởng thiết kế hợp lý, tránh việc vội vàng muốn giải quyết vấn đề mà để lại hậu quả nghiêm trọng về sau.