Numéro Sciper	PARTNER SCIPER	Architect. (3pts max)	Architecture violation comment	Class encapsul. (4pts max)	Class violation comment
270354	302076	2	{A1}: only one fonction of simulation module must be called by main. Warning: ask permission to Mr. Boulic before using additional modules (in this case, renaming map as obstacles)	1,5	[C1]: public attricute ERROR_MODE_KEYWORD in class simulation (in fact, you don't need to have this attribute, comparing argv[2] with just the string "Error" is not a mistake, or as an alternative you can move his definition in the main. Ideally, if you rellay want to keep it, make it const std::string). [C2]: all your constructors have their initialisation list in the .h
271829	296759	2	[A1] projet.cc includes ball, player, etc	3	[C1] nbr_coin_carre (map.h)
273712	301881	3	Bon travail sur l'architecture du projet !	4	La modularisation est bien réalisée elle aussi.
275336	289067	3		4	Note: you can limit the use of static variables by handling lists of a certain type at a higher level as attributes (e.g. balls would be an attribute of the class Simulation)
276861	301291	3	Très bon respect de l'architecture ! Continuez	4	Bon travail ! Remarque : Les modules player,ball et map sont peut-être un peu modestes par rapport à simulation mais c'est un choix toléré.
279202	288530	3	Parfait	4	Parfait
282211	288820	3	You could have put all the file reading in the simulation module	4	warning: too many static variables
282287	283419	1	[A1] project.cc does all the work of simulation. [A2] simulation module does not exist.	0	[C1] ball_x, ball_y, angle, init_balls, collision_b_b, ... (rien n'est privé, aucune méthode n'est dans la classe)

282360	289256	3	ok	3,5	[C2] Point, Rectangle, Vecteur
282409	284529	3		4	Warning: constructors must be implemented in the source file (convention)
282416	287609	3	ok	4	ok
282435	286955	2	[A2] : Simulation ne voit Player, Ball et Obstacle que parce qu'il voit Map, qui lui-même voit (inutilement) Player, Obstacle et Ball.	4	Warning : Projet, Map contiennent des includes inutiles. De plus, il y a des includes dans les .cc, et qui répètent parfois ceux du .h. Pour finir, vous utilisez trop de variables static, notamment à des moments où elles ne sont pas justifiées, comme lorsque ces variables sont des attributs privés d'une classe. Cela peut réduire les performances de votre programme. Attention à bien comprendre ce qu'implique static.
282515	288166	3	Très bonne architecture programme et respect du Modèle. Le module tools pourrait être plus complet et faciliter la gestion d'entités "Cercle" par exemple.	3	La modularisation est bonne cependant deux points sont à noter : - Vous pourriez utiliser plus de méthodes qu'uniquement getters, setters et constructeur. Il s'agit d'un outil très pratique et puissant. - La définition de vos méthodes doit absolument être externalisée (pour avoir une interface propre). La seule exception admise est celle des getters mais seulement si la definition est sur la même ligne que le prototype. D'ou l'erreur [C2] map.h, ball.h
282565	284284	3	Ok. Attention, des include sont inutiles dans le fichier projet.cc	4	Très bien.
282657	287800	3	Okay. Warning : you use a lot of static variables inside your method decodage_ligne. This might be good to change a bit the design of this method.	4	Okay. Warning : distance method in tools.h clearly belong to the Point class as it computes only the distance of Point.
282708	288602	2	There is no utility module... "tools"	4	

282756	286734	3	Good	4	Good
282836	302583	2	You should not include all files in all the files ! (Especially if you don't use them ...)	4	Good
282844	288453	3	Good	4	Good. Just try to avoid putting "using namespace std;" in header file, as it can cause inclusion problem in some situation
282988	284213	3	Parfait	3	[C2]:ball.h: il y a des fonctions qui devraient être dans la classe qui ont été sortie de cette dernière. Mettez-les dedans !
283111	284430	3	ok. warnings: 1)tools fait très peu de choses. 2) A qui sont destinées les fonction visibles dans simulation.h ?	4	ok, warning: les fichiers en-têtes contiennent des #include inutiles
283183	296952	3	Nice job! Make sure to use the function distance that you have defined in tools	4	Nice job!
283196	284316	3	Très bien. Le module tools pourrait être completé.	4	Ok. Vous devriez cependant revoir vos structures de données (vectors de players, balls et obstacles) : les avoir en variables locales d'une fonction ne semble pas être une option adaptée pour la suite du projet.
283391	289122	3	Good	4	Okay. Warning : you put an unused inclusion of error.h in tools.h. This should be removed. Also you should not put a "using namespace std" in the headers file.
283395	288243	3	Warning : dans le cas d'ajout de modules, bien penser à en justifier la raison dans le fichier .h à l'aide de commentaires. Si certains peuvent être justifiés (par exemple obstacle), d'autres modules auraient plus leur place au sein d'autres modules déjà existants (par exemple, les collisions pourraient être traitées entre simulation et tools et info_file pourrait faire partie de simulation). Au delà de ça, vous semblez bien avoir compris l'architecture du projet, c'est bien !	2,5	[C2] ball.h, obstacle.h, player.h : Attention, toutes les méthodes doivent être externalisées dans le fichier .cc correspondant, y-compris les constructeurs. Sinon, dans l'ensemble, bon travail sur la modularisation.
283419	282287	1	[A1] project.cc does all the work of simulation. [A2] simulation module does not exist.	0	[C1] ball_x, ball_y, angle, init_balls, collision_b_b, ... (rien n'est privé, aucune méthode n'est dans la classe)

283442	287811	3	Très bon travail, l'architecture du projet est bien respectée	2,5	[C2]ball.h, map.h, player.h : Attention, toutes les méthodes doivent être externalisées dans le fichier .cc correspondant, y-compris les constructeurs. Sinon, dans l'ensemble, bon travail sur la modularisation.
283470	286317	3	Ok. Les méthodes de collision dans player, ball et obstacle pourraient renvoyer un booléen et être utilisées pour les messages d'erreur dans simulation (toujours dans des fonctions)	4	Ok.
283616	301494	3	Okay. Warning : You will need to ask Ronan Boulic to add the moduleLec. However, this might not necessarily be a bad idea (even if the name could be more subtle)	4	Good. You might just want to think whether distanceVectorielle should really be outside the class Point.
283652	303039	3	Good	2,5	[C2] tools.h constructor, player.h, map.h headers file should only contains method prototype (or exceptionally definition on the same line as the prototype)
283670	286570	2	[A3] Tool should not even have an idea of what a player is... (tools.h 60) It is supposed to be a low-level utility module	4	Warning: constructors must be implemented in the source file (convention)
283792	287054	3		4	
283796	286386	2	[A1] : le module projet ne doit voir que le module simulation, vous ne devez pas inclure Player.h	4	Les principes d'encapsulation sont bien respectés. Attention toutefois, vos vecteurs statiques devraient être définies comme attributs privés des classes correspondantes afin de respecter parfaitement les convention Orienté- Objet
284024	289159	3	No architecture errors	0	[C1] ball.cc has 3 global variables , map.cc has 1 global variables, player.cc has 3 global variables
284194	287056	3	Why do you often repeat in .cc the function prototypes found in the respective .h?	4	In the future put the static variables in player, map and ball as private static attributes.
284196	299245	3		4	
284213	282988	3	Parfait	3	[C2]:ball.h: il y a des fonctions qui devraient être dans la classe qui ont été sortie de cette dernière. Mettez-les dedans !

284284	282565	3	Ok. Attention, des include sont inutiles dans le fichier projet.cc	4	Très bien.
284316	283196	3	Très bien. Le module tools pourrait être completé.	4	Ok. Vous devriez cependant revoir vos structures de données (vectors de players, balls et obstacles) : les avoir en variables locales d'une fonction ne semble pas être une option adaptée pour la suite du projet.
284402	289234	3		4	
284430	283111	3	ok. warnings: 1)tools fait très peu de choses. 2) A qui sont destinées les fonction visibles dans simulation.h ?	4	ok, warning: les fichiers en-têtes contiennent des #include inutiles
284529	282409	3		4	
284564	289285	3	Well Modularized!	4	warning: too many static variables
284867	296833	3	Parfait	4	Parfait, essayez de ne pas abuser de retour à la ligne pour vos définitions de constructeurs mais il n'y a sinon rien à redire !
284888	286950	2	[A2]: player.cc includes simulation.h	0	[C1]: ball.cc and player.cc have global variables speed, radius
284896	301437	3	Warning : dans l'état actuel il semble que lecture est un module différent de simulation, car il est appelé à part puis la simulation est appelée. Une meilleure architecture serait de fournir le nom de fichier à Simulation qui se charge de lire le fichier puis d'exécuter les erreurs.	4	L'encapsulation est bien comprise. Warning: Expliquez dans les commentaires lorsque vous créez un module supplémentaire, comme test ou lecture.
285416	289276	2	[A3] Tools was supposed to be independant from higher module.	3	[C2] You should externalize all the definition of your method in your .cc (player.h, map.h, ball.h

286317	283470	3	Ok. Les méthodes de collision dans player, ball et obstacle pourraient renvoyer un booléen et être utilisées pour les messages d'erreur dans simulation (toujours dans des fonctions)	4	Ok.
286386	283796	2	[A1] : le module projet ne doit voir que le module simulation, vous ne devez pas inclure Player.h	4	Les principes d'encapsulation sont bien respectés. Attention toutefois, vos vecteurs statiques devraient être définies comme attributs privés des classes correspondantes afin de respecter parfaitement les convention Orienté- Objet
286570	283670	2	[A3] Tool should not even have an idea of what a player is... (tools.h 60) It is supposed to be a low-level utility module	4	Warning: constructors must be implemented in the source file (convention)
286641	286721	3	Warning : dans le cas d'ajout de modules, bien penser à en justifier la raison dans le fichier .h à l'aide de commentaires. Votre module parameters me semble toutefois bien, vous pourriez d'ailleurs éventuellement le fusionner avec define. Sinon, bon travail sur l'architecture, c'est bien !	4	La modularisation et l'externalisation sont bien réalisées, c'est bien !
286721	286641	3	Warning : dans le cas d'ajout de modules, bien penser à en justifier la raison dans le fichier .h à l'aide de commentaires. Votre module parameters me semble toutefois bien, vous pourriez d'ailleurs éventuellement le fusionner avec define. Sinon, bon travail sur l'architecture, c'est bien !	4	La modularisation et l'externalisation sont bien réalisées, c'est bien !
286734	282756	3	Good	4	Good
286830	289281	3	Ok	4	Parfait, encapsulation bien réalisée.
286950	284888	2	[A2]: player.cc includes simulation.h	0	[C1]: ball.cc and player.cc have global variables speed, radius

286955	282435	2	[A2] : Simulation ne voit Player, Ball et Obstacle que parce qu'il voit Map, qui lui-même voit (inutilement) Player, Obstacle et Ball.	4	Warning : Projet, Map contiennent des includes inutiles. Par ailleurs, vous utilisez trop de variables static, notamment à des moments où elles ne sont pas justifiées, comme lorsque ces variables sont des attributs privés d'une classe. Cela peut réduire les performances de votre programme. Attention à bien comprendre ce qu'implique static.
286962	301464	2	[A2]Le module tool est bien vide ! Vous n'avez pas assez délégué les fonctions liées à la géométrie pour hélas la traiter dans balle et dans joueur. Ce qui est faux. A part cela, vous avez créé un fichier inutile define2.h ; son contenu peut très bien aller ailleurs.	4	Correct. Attention avec les endif des ifndef dans les .h. Il faut les mettre tout à la fin du fichier
287015	296048	2	[A1]projet.cc : ce module n'est sensé gérer que argc et argv lors du rendu 1 : le reste des actions doit être délégué à simulation. Du reste, l'architecture du projet reste bien respectée.	4	La modularisation et l'externalisation sont elles par contre bien respectées tout du long, c'est bien !
287038	297846	3	Perfect!	4	Nice job!
287054	283792	3		4	
287056	284194	3	Why do you often repeat in .cc the function prototypes found in the respective .h?	4	In the future put the static variables in player, map and ball as private static attributes.
287173	301560	3	L'architecture du projet est bien respectée, bon travail. Toutefois, je pense qu'il serait mieux d'étoffer votre module tool avec des concepts tels que Cercle par exemple, au nom du principe de réutilisation et d'abstraction. Egalement dans tool, pourquoi avoir Point et Vector alors que les structures sont parfaitement identiques ?	4	L'externalisation et l'encapsulation sont bien respectés, bravo !
287327	287388	3	Nice job!	4	Looks good!
287388	287327	3	Nice job!	4	Looks good!

287433	300540	3	Well Modularized!	4	warning: too many static variables
287459	302017	3	Excellent respect de l'architecture ! RAS	4	Très bonne modularisation et encapsulation. Les modules player,ball et map sont peut-être un peu modestes par rapport à simulation mais c'est un choix toléré.
287480					
287609	282416	3	ok	4	ok
287725	288407	3	Good job!	4	Well done!
287759	295954	3	L'architecture du projet est bien respectée	4	Bon travail sur l'encapsulation et la modularisation
287800	282657	3	Okay. Warning : you use a lot of static variables inside your method decodage_ligne. This might be good to change a bit the design of this method.	4	Okay. Warning : distance method in tools.h clearly belong to the Point class as it computes only the distance of Point.
287811	283442	3	Très bon travail, l'architecture du projet est bien respectée	2,5	[C2]ball.h, map.h, player.h : Attention, toutes les méthodes doivent être externalisées dans le fichier .cc correspondant, y-compris les constructeurs. Sinon, dans l'ensemble, bon travail sur la modularisation.
288166	282515	3	Très bonne architecture programme et respect du Modèle. Le module tools pourrait être plus complet et faciliter la gestion d'entités "Cercle" par exemple.	3	La modularisation est bonne cependant deux points sont à noter : - Vous pourriez utiliser plus de méthodes qu'uniquement getters, setters et constructeur. Il s'agit d'un outil très pratique et puissant. - La définition de vos méthodes doit absolument être externalisée (pour avoir une interface propre). La seule exception admise est celle des getters mais seulement si la definition est sur la même ligne que le prototype. D'ou l'erreur [C2] map.h, ball.h

288243	283395	3	Warning : dans le cas d'ajout de modules, bien penser à en justifier la raison dans le fichier .h à l'aide de commentaires. Si certains peuvent être justifiés (par exemple obstacle), d'autres modules auraient plus leur place au sein d'autres modules déjà existants (par exemple, les collisions pourraient être traitées entre simulation et tools et info_file pourrait faire partie de simulation). Au delà de ça, vous semblez bien avoir compris l'architecture du projet, c'est bien !	2,5	[C2] ball.h, obstacle.h, player.h : Attention, toutes les méthodes doivent être externalisées dans le fichier .cc correspondant, y-compris les constructeurs. Sinon, dans l'ensemble, bon travail sur la modularisation.
288252	303011	3	L'architecture est bien respectée	4	Malgré les bémols qui suivent, les grands principes d'encapsulation sont bien respectés. Warning: vous définissez vos vecteurs comme globaux au module qui les contient. Il semble plus logique de placer ces variables statique dans la partie private de votre classe pour en faire des attributs bien encapsulés. Warning : projet.cc contient des includes inutiles. Warning : toutes vos classes sont définies avec des fonctions à part, dans un style purement séquentiel. Or ces fonctions devraient être des méthodes des classes, c'est une violation du principe d'Orienté-objet. Vous devez corriger ça pour le futur.
288287	299554	3	Les principes de l'architecture sont bien compris, bravo	4	Warning: ces trois variables static du module Simulation pourraient être évitées de bien des manières. Essayez d'imaginer une autre solution. L'encapsulation est plutot bien faite sinon
288312	296165	3	Ok. Attention aux includes inutiles dans projet.cc.	4	Très bien. Pourquoi utiliser une variable statique et non pas une macropour margeLecture dans simulation.cc ?
288407	287725	3	Good job	4	Well done!
288453	282844	3	Good	4	Good. Just try to avoid putting "using namespace std;" in header file, as it can cause inclusion problem in some situation
288530	279202	3	Parfait	4	Parfait

288580	288825	3	Ok.	4	Ok.
288602	282708	2	There is no utility module... "tools"	4	
288820	282211	3	You could have put all the file reading in the simulation module	4	warning: too many static variables
288825	288580	3	Ok.	4	Ok.
288838	289028	1	[A1] main doit juste lancer la simulation [A2] Il faut un module simulation responsable de superviser le comportement des player, ball et map . Vos .h sont très chargés. Eventuellement définissez vos variables static directement dans le .cc avec les méthodes pour les manipuler hors de l'interface de la classe (mais toujours avec leur définitions dans le .cc).	4	Ok. Le destructeur par défaut ne fait déjà rien donc inutile de le redéfinir.
289028	288838	1	[A1] main doit juste lancer la simulation [A2] Il faut un module simulation responsable de superviser le comportement des player, ball et map . Vos .h sont très chargés. Eventuellement définissez vos variables static directement dans le .cc avec les méthodes pour les manipuler hors de l'interface de la classe (mais toujours avec leur définitions dans le .cc).	4	Ok. Le destructeur par défaut ne fait déjà rien donc inutile de le redéfinir.
289067	275336	3	Copy	4	
289121	296142	3	Très bon respect de l'architecture ! Continuez	4	Bon travail ! Remarque : Les modules player,ball et map sont peut-être un peu modestes par rapport à simulation mais c'est un choix toléré.
289122	283391	3	Good	4	Okay. Warning : you put an unused inclusion of error.h in tools.h. This should be removed. Also you should not put a "using namespace std" in the headers file.

289159	284024	3	No architecture errors	0	[C1] ball.cc has 3 global variables , map.cc has 1 global variables, player.cc has 3 global variables
289188	295855	3	Okay. Warning : you should remove unused inclusion that are in projet.cc and tools.h !	3,5	[C2] you should not put constructor definition in header files
289234	284402	3		4	
289256	282360	3	ok	3,5	[C2] Point, Rectangle, Vecteur
289276	285416	2	[A3] Tools was supposed to be independant from higher module.	3	[C2] You should externalize all the definition of your method in your .cc (player.h, map.h, ball.h
289281	286830	3	Ok	4	Parfait, encapsulation bien réalisée.
289285	284564	3	Well Modularized!	4	warning: too many static variables
289304	295808	3	L'architecture est correcte. Warning : Vos includes sont quasiment tous mal écrits, car vous ne laissez pas d'espace entre la directive include et le nom du fichier à inclure.	4	Warning : les prototypes de méthodes au sommet de simulation.cpp devraient être définis dans la partie private de simulation.h : même effet mais plus propre. Encapsulation bien comprise sinon.
289324	289596	3		4	
289596	289324	3		4	

290067	300669	2	Dans l'ensemble, vous semblez avoir bien compris le rôle de l'architecture (séparation des modules...). Cependant, même si cela part d'une bonne intention, votre architecture ne respecte pas celle de la donnée du projet (ou est simulation ? pourquoi input ?). Il est toléré de créer de nouveaux modules mais seulement si ces derniers sont bien documentés et justifiés dans le projet. Warning : "define.h" ne doit absolument pas être inclus dans tools.h !! Tools est un module de bas niveau qui ne doit pas travailler avec des structure liées au projet. Aussi, je trouve que votre module Tools est assez modeste, cela vous simplifierait la vie si vous y créez des structures Cercle, Point... et des méthodes travaillant dessus.	4	Vos classes sont parfaitement définies et documentées . Bravo !
290470	296032	2	[A3] Tools ne devrait pas avoir besoin de nbCells qui est une entité du module Simulation. Cela brise un point important des contraintes architecturales du projet	4	Warning : Ball.cc: 7, Player.cc: 7, Map.cc: 9, Simulation.cc: 15. Afin de parfaitement respecter le principe d'encapsulation, ces variables statiques devraient être des attributs privés des classes correspondantes. Warning : vous définissez des fonctions à côtés de classes, alors même que ces fonctions devraient être des méthodes de ces classes ! Il faut corriger cela.
295758	296442	3	Good	4	Good
295784	296046	2	[A1] main.cc should only check command line arguments and not perform any test (and here it doesn't even chech command line arguments... it literally should call only one function in simulation and that's all for the rendu1)	1	[C2]: all your constructors have their initialisation list in the .h
295808	289304	3	L'architecture est correcte. Warning : Vos includes sont quasiment tous mal écrits, car vous ne laissez pas d'espace entre la directive include et le nom du fichier à inclure.	4	Warning : les prototypes de méthodes au sommet de simulation.cpp devraient être définis dans la partie private de simulation.h : même effet mais plus propre. Encapsulation bien comprise sinon.

295821	299895	3	Architecture particulièrement bien respectée, très bon travail ! Toutefois, dans tool, il ne me paraît pas adapté de mettre la norme en attribut pour le vecteur ou la longueur pour le segment (etc.) : privilégiez l'implémentation de méthodes les calculant à la volée et les retournant sans les enregistrer dans la structure ; selon votre méthode, si le vecteur par exemple est changé, rien n'assure que la valeur associée à norme sera elle aussi mise à jour systématiquement.	4	Bon travail sur le respect de l'encapsulation et de l'externalisaton.
295855	289188	3	Okay. Warning : you should remove unused inclusion that are in projet.cc and tools.h !	3,5	[C2] you should not put constructor definition in header files
295890	302360	3	Ok.	3,5	[C2] tool.h. Les constructeurs doivent être définis dans le .cc
295927	300882	3	Parfait	4	Parfait
295954	287759	3	L'architecture du projet est bien respectée	4	Bon travail sur l'encapsulation et la modularisation
296023	296815	3	Ok	4	Très bien. Principes d'encapsulation et modularisation bien mis en oeuvre.
296026	301418	3		4	Warning: constructors must be implemented in the source file (convention) There seem to be a confusion about the use of the static keyword... Unless I'm missing something, it's basically meaningless declaring a stating variable inside the main function... Ask an assistant if you can't see why! tools.cc 6 where is "tools_que_jePeuxUtiliser.h" ?

296032	290470	2	[A3] Tools ne devrait pas avoir besoin de nbCells qui est une entité du module Simulation. Cela brise un point important des contraintes architecturales du projet	4	Warning : Ball.cc: 7, Player.cc: 7, Map.cc: 9, Simulation.cc: 15. Afin de parfaitement respecter le principe d'encapsulation, ces variables statiques devraient être des attributs privés des classes correspondantes. Warning : vous définissez des fonctions à côtés de classes, alors même que ces fonctions devraient être des méthodes de ces classes ! Il faut corriger cela.
296046	295784	2	[A1] main.cc should only check command line arguments and not perform any test (and here it doesn't even chech command line arguments... it literally should call only one function in simulation and that's all for the rendu1)	1	[C2]: all your constructors have their initialisation list in the .h
296048	287015	2	[A1]projet.cc : ce module n'est sensé gérer que argc et argv lors du rendu 1 : le reste des actions doit être délégué à simulation. Du reste, l'architecture du projet reste bien respectée.	4	La modularisation et l'externalisation sont elles par contre bien respectées tout du long, c'est bien !
296050	297097	3	Parfait, vous semblez avoir bien compris comment le Model_View_Controller fonctionne et appliquez à la lettre les principes de la programmation orientée objet. Warning : Même si la classe me semble bonne, demandez à Mr Boulic l'autorisation d'utiliser la classe Actor.	4	Excellente modularisation et gestion de classes. Remarque : On a souvent l'impression que des fonctions sont copiées-collées (RetrieveNbXXX par exemple), pensez à la réutilisation de code (copier-coller n'est pas très safe)
296057	301225	3	Très bien	4	Parfait, principe d'encapsulation très bien mis en oeuvre.
296093	300411	3	Très bien	4	Parfait, encapsulation très bien respectée et module tools très complet.
296142	289121	3	Très bon respect de l'architecture ! Continuez	4	Bon travail ! Remarque : Les modules player,ball et map sont peut-être un peu modestes par rapport à simulation mais c'est un choix toléré.
296165	288312	3	Ok. Attention aux includes inutiles dans projet.cc.	4	Très bien. Pourquoi utiliser une variable statique et non pas une macropour margeLecture dans simulation.cc ?
296169	302659	3	Excellent respect de l'architecture du module projet et du Modèle. Warning : Vous avez oublié de changer le nom de classe "Obstacle" dans tools. J'ai pris en compte le commentaire pour cette fois.	4	Parfait ! Pensez peut-être à utiliser la puissance des méthodes (accès aux attributs private) et de leur encapsulation lorsque c'est possible plutot que de définir des fonctions hors classe

296235	297075	3	Perfect	4	Good job
296372	303028	3	Très bien	4	Très bien. Est-il nécessaire de déclarer des constantes dans projet.cc alors que celles-ci ne sont utilisées qu'une seule fois chacune, et ne sont pas amenées à évoluer ?
296399	296487	3		4	
296442	295758	3	Good	4	Good
296487	296399	3		4	
296643	299689	3	Très bien	4	Ok. Warning : Vous auriez pu éviter le recours à autant de variables statiques en modifiant légèrement la structure de votre code.
296644	298386	2	[A3] le module tools doit rester indépendant du projet : on applique une "contrainte sémantique importante". Or votre classe Case utilise des paramètres comme nbCell et sont directement liés à notre projet (haut niveau). Une classe Square ou Rect avec des coordonnées x et y serait plus appropriée. Voir p6 de la donnée. Cependant, bravo pour votre décomposition et le respect de l'architecture !	4	Parfait !
296759	271829	2	[A1] projet.cc includes ball, player, etc	3	[C1] nbr_coin_carre (map.h)
296815	296023	3	Ok	4	Très bien. Principes d'encapsulation et modularisation bien mis en oeuvre.
296816	301033	3		4	

296827	299894	3		4	
296833	284867	3	Parfait	4	Parfait, essayez de ne pas abuser de retour à la ligne pour vos définitions de constructeurs mais il n'y a sinon rien à redire !
296867	300131	3	Good job!	4	Well done!
296935	301358	3	No architecture errors	4	Perfect encapsulation
296952	283183	3	Nice job! Make sure to use the function distance that you have defined in tools	4	Nice job!
297075	296235	3	Perfect	4	Good job
297097	296050	3	Parfait, vous semblez avoir bien compris comment le Model_View_Controller fonctionne et appliquez à la lettre les principes de la programmation orientée objet. Warning : Même si la classe me semble bonne, demandez à Mr Boulic l'autorisation d'utiliser la classe Actor.	4	Excellente modularisation et gestion de classes. Remarque : On a souvent l'impression que des fonctions sont copiées-collées (RetrieveNbXXX par exemple), pensez à la réutilisation de code (copier-coller n'est pas très safe)
297151	301452	3	Ok. Peut-être ajoutez des méthodes afin de simplifier l'écriture de vos méthodes de collision. Par exemple une méthode qui fait le traitement entre un élèment de chaque et qui ensuite seraient appellées sur vos listes (e.g collisionBallPlayer(Player p,...)).	4	Ok.

297176	301998	3	Excellent travail sur l'architecture, bravo ! Attention cependant à ne pas fuir l'orienté-objet, vu votre niveau une classe Simulation regroupant la lecture, le test des erreurs et la création des instances aurait été parfait ! Warning : il est important de justifier en commentaire le besoin que vous avez eu de créer les modules lecture et detectionErreur, bien qu'on devine aisément qu'ils sont sous-modules de Simulation	4	L'encapsulation est correcte et bien comprise. Il est toutefois inutile de garder une variable à jour censée contenir le nombre de Player ou Ball en jeu : un simple appel à vector.size() vous retourne cette info et vous évite de potentielles erreurs dues à un oubli de maintien d'une variable compteuse. Warning : il faut corriger vos includes : on inclut QUE dans le .h, et le .cc contient uniquement son propre .h en include. Ball.cc n'a donc en include que "ball.h"
297846	287038	3	Perfect!	4	Nice job!
298011	302747	3		4	Be careful with static variables in modules. Some possibilites to use them less: When you declare a list of As in the module where you define A, you could also handle this list in a "higher" module (e.g. list of balls in simulation) For your static variables in simulation, you may want to encompass a whole context of the simulation in a class, and those variables could become attributes.
298386	296644	2	[A3] le module tools doit rester indépendant du projet : on applique une "contrainte sémantique importante". Or votre classe Case utilise des paramètres comme nbCell et sont directement liés à notre projet (haut niveau). Une classe Square ou Rect avec des coordonnées x et y serait plus appropriée. Voir p6 de la donnée. Cependant, bravo pour votre décomposition et le respect de l'architecture !	4	Parfait !
299245		3		4	
299479	299496	3		4	Warning: constructors must be implemented in the source file (convention)
299496	299479	3		4	Warning: constructors must be implemented in the source file (convention)

299554	288287	3	Les principes de l'architecture sont bien compris, bravo	4	Warning: ces trois variables static du module Simulation pourraient être évitées de bien des manières. Essayez d'imaginer une autre solution. L'encapsulation est plutot bien faite sinon
299561	300845	3	Parfait	4	Parfait, class de la simulation très lisible
299689	296643	3	Très bien	4	Ok. Warning : Vous auriez pu éviter le recours à autant de variables statiques en modifiant légèrement la structure de votre code.
299724	301959	2	Warning: ask Mr. Boulic if you can use additional modules (in this case, world) Warning [A1]: main.cc is not supposed to include error.h	4	Perfect encapsulation
299882	301272	3	Good	2,5	[C2] Class should go in header file ! Only the implementation of methods should go in the .cc (for class Ball, Player, Map)
299892	300594	2	L'architecture est bien maitrisée et appliquée ! Bravo. Attention tout de même au module Tools qui doit rester de bas-niveau et dans lequel on ne devrait surtout pas retrouver la notion de "DIM_MAX" venant de defintions.h	4	Parfait. Interfaces très propres et épurées. Warning : La modularisation est très très partielle, 80% du programme est concentré dans simulation et les autres modules ne semblent être que des prétextes. Vous pourriez les utiliser plus efficacement et avoir une meilleure modularisation.
299894	296827	3		4	
299895	295821	3	Architecture particulièrement bien respectée, très bon travail ! Toutefois, dans tool, il ne me paraît pas adapté de mettre la norme en attribut pour le vecteur ou la longueur pour le segment (etc.) : privilégiez l'implémentation de méthodes les calculant à la volée et les retournant sans les enregistrer dans la structure ; selon votre méthode, si le vecteur par exemple est	4	Bon travail sur le respect de l'encapsulation et de l'externalisaton.
300131	296867	3	Good job!	4	Well done!
300253	302703	3	Good job!	4	Perfect

300411	296093	3	Très bien	4	Parfait, encapsulation très bien respectée et module tools très complet.
300443	301030	3	Bon respect de l'architecture : on voit que vous avez compris le concept.	3	[C1]simulation.h:26-28 : attention ! Strictement tous les attributs doivent être privés. Utilisez plutôt des getters et setters si besoin est. Sinon, l'externalisation est bien respectée.
300456	301829	3	Architecture nickel	4	L'encapsulation est parfaite et bien comprise
300540	287433	3	Well Modularized!	4	warning: too many static variables
300558	302896	3	Les principes de l'architecture sont compris mais quelle machine de guerre que projet ! Il est inutile d'aller aussi loin, un simple argv[1] == "Error" est bien suffisant, avec à la rigueur quelques checks pour s'assurer qu'un input faux ne peux faire planter le programme.	4	Le principe d'encapsulation est bien respecté. Warning : vous faites des includes dans les .cc, ils sont censé se trouver dans les .h, le .cc ne contenant que l'include vers le .h correspondant. Warning : un prototype rendu visible dans le .h ne doit pas se trouver également au sommet du .cc.
300559	302674	3	Warning: ask Mr. Boulic if you can use additional modules (in this case, setup)	2	[C1]: ball.cc and player.cc have global attribute rayon
300594	299892	2	L'architecture est bien maitrisée et appliquée ! Bravo. Attention tout de même au module Tools qui doit rester de bas-niveau et dans lequel on ne devrait surtout pas retrouver la notion de "DIM_MAX" venant de defintions.h	4	Parfait. Interfaces très propres et épurées. Warning : La modularisation est très très partielle, 80% du programme est concentré dans simulation et les autres modules ne semblent être que des prétextes. Vous pourriez les utiliser plus efficacement et avoir une meilleure modularisation.
300615	301581	3	Ok. simulation.h : expliquer pourquoi vous incluez d'autres modules tels que obstacle.	4	Cobstacle, si le constructeur est vide rester cohèrent avec les autres constructeurs vides. Ne faites pas commencer le nom des classes par C (e.g. Cobstacle -> Obstacle)
300620	301641	3	ok	4	ok

300669	290067	2	Dans l'ensemble, vous semblez avoir bien compris le rôle de l'architecture (séparation des modules...). Cependant, même si cela part d'une bonne intention, votre architecture ne respecte pas celle de la donnée du projet (ou est simulation ? pourquoi input ?). Il est toléré de créer de nouveaux modules mais seulement si ces derniers sont bien documentés et justifiés dans le projet. Warning : "define.h" ne doit absolument pas être inclus dans tools.h !! Tools est un module de bas niveau qui ne doit pas travailler avec des structure liées au projet. Aussi, je trouve que votre module Tools est assez modeste, cela vous simplifierait la vie si vous y créez des structures Cercle, Point... et des méthodes travaillant dessus.	4	Vos classes sont parfaitement définies et documentées . Bravo !
300803	301708	2	[A2]Vous avez rajouter de nombreux fichiers, alors qu'il était simplement possible de mettre leur contenu dans d'autres fichiers comme simulation	4	Parfait
300845	299561	3	Parfait	4	Parfait, class de la simulation très lisible
300882	295927	3	Parfait	4	Parfait
301001	302551	2	[A2]simulation: Bonne implémentation mais votre module simulation fait la quasi-totaltié du module à sa place: déléguer correctment	4	Parfait
301030	300443	3	Bon respect de l'architecture : on voit que vous avez compris le concept.	3	[C1]simulation.h:26-28 : attention ! Strictement tous les attributs doivent être privés. Utilisez plutôt des getters et setters si besoin est. Sinon, l'externalisation est bien respectée.
301033	296816	3		4	

301225	296057	3	Très bien	4	Parfait, principe d'encapsulation très bien mis en oeuvre.
301265	301403	2	[A2] module Data casse l'isolation des modules	4	
301272	299882	3	Good	2,5	[C2] Class should go in header file ! Only the implementation of methods should go in the .cc (for class Ball, Player, Map)
301291	276861	3	Très bon respect de l'architecture ! Continuez	4	Bon travail ! Remarque : Les modules player,ball et map sont peut-être un peu modestes par rapport à simulation mais c'est un choix toléré.
301358	296935	3	No architecture errors	4	Perfect encapsulation
301403	301265	2	[A2] module Data casse l'isolation des modules	4	
301411	302276	3		4	Warning: constructors must be implemented in the source file (convention) There seem to be quite a lot of static variables : using a class for a simulation context, the static variables of this module would simply become attributes, and the lists of instances of other classes could be stored at this higher level too, within this simulation class (instead of within the modules themselves)
301418	296026	3		4	Warning: constructors must be implemented in the source file (convention) There seem to be a confusion about the use of the static keyword... Unless I'm missing something, it's basically meaningless declaring a stating variable inside the main function... Ask an assistant if you can't see why! tools.cc 6 where is "tools_que_jePeuxUtiliser.h" ?

301437	284896	3	Warning : dans l'état actuel il semble que lecture est un module différent de simulation, car il est appelé à part puis la simulation est appelée. Une meilleure architecture serait de fournir le nom de fichier à Simulation qui se charge de lire le fichier puis d'exécuter les erreurs.	4	L'encapsulation est bien comprise. Warning: Expliquez dans les commentaires lorsque vous créez un module supplémentaire, comme test ou lecture.
301452	297151	3	Ok. Peut-être ajoutez des méthodes afin de simplifier l'écriture de vos méthodes de collision. Par exemple une méthode qui fait le traitement entre un élèment de chaque et qui ensuite seraient appellées sur vos listes (e.g collisionBallPlayer(Player p,...)).	4	Ok.
301464	286962	2	[A2]Le module tool est bien vide ! Vous n'avez pas assez délégué les fonctions liées à la géométrie pour hélas la traiter dans balle et dans joueur. Ce qui est faux. A part cela, vous avez créé un fichier inutile define2.h ; son contenu peut très bien aller ailleurs.	4	Correct. Attention avec les endif des ifndef dans les .h. Il faut les mettre tout à la fin du fichier
301494	283616	3	Okay. Warning : You will need to ask Ronan Boulic to add the moduleLec. However, this might not necessarily be a bad idea (even if the name could be more subtle)	4	Good. You might just want to think whether distanceVectorielle should really be outside the class Point.
301560	287173	3	L'architecture du projet est bien respectée, bon travail. Toutefois, je pense qu'il serait mieux d'étoffer votre module tool avec des concepts tels que Cercle par exemple, au nom du principe de réutilisation et d'abstraction. Egalement dans tool, pourquoi avoir Point et Vector alors que les structures sont parfaitement identiques ?	4	L'externalisation et l'encapsulation sont bien respectés, bravo !
301581	300615	3	Ok. simulation.h : expliquer pourquoi vous incluez d'autres modules tels que obstacle.	4	Cobstacle, si le constructeur est vide rester cohèrent avec les autres constructeurs vides. Ne faites pas commencer le nom des classes par C (e.g. Cobstacle -> Obstacle)
301595	302759	1	[A2]Vous ne déléguez pas assez aux modules ball et player. Vous les utilisez comme simples espaces de stockage. [A3]Le module tool travaille sur des joueurs et des balles alors qu'elles doivent lui être opaques.	2,5	[C2]Constructeur obstacle, Constucteur Balle, Constructeur Player

301633	302909	3	Ok. Vu que vous utilisez une hierarchie de classes avec Cercle les méthodes de collisions peuvent être mises directement dans la classe. Cela permettra aussi de spécialiser le comportement en cas de collisions. Ce qui, pour l'instant n'est pas beaucoup exploité par votre code. Repensez à la relation "a un" et "est un".	4	Ok.
301641	300620	3	ok	4	ok
301708	300803	2	[A2]Vous avez rajouter de nombreux fichiers, alors qu'il était simplement possible de mettre leur contenu dans d'autres fichiers comme simulation	4	Parfait
301750	302031	3	Parfait	4	Parfait
301766	302289	3	Warning : dans le cas d'ajout de modules, bien penser à en justifier la raison dans le fichier .h à l'aide de commentaires. Si des modules tels qu'obstacle me paraissent justifiés, d'autres me paraissent moins nécessaires (sans être indésirables non plus), tels que leture ou tests qui auraient leur place au sein de simulation ou des modules correspondant aux divers éléments du jeu, module qui pour l'instant est un peu vide. Dans tous les cas, si vous ajoutez des modules => justifiez votre choix dans les commentaires des .h respectifs.	4	Bon travail pour la modularisation et l'externalisation
301829	300456	3	Architecture nickel	4	L'encapsulation est parfaite et bien comprise
301836	303153	3	warning : justifiez l'utilisation des .h autres que ball,player et map dans simulation.h. Tentez de déleguer plus aux autres classes et mettez moins de code dans simulation.cc.	4	Ok
301881	273712	3	Bon travail sur l'architecture du projet !	4	La modularisation est bien réalisée elle aussi.

301949	302565	3	Ok.	4	Ok. Pas besoin de déclarer le destructeur si vous ne le redéfinissez pas. Pour éviter les interfaces trop chargés choisissez bien ce à quoi l'utilisateur a accès. Certaines méthodes d'aides peuvent être définies dans le .cc sans les déclarer dans le .h.
301956	301994	3	Warning: ask Mr. Boulic if you can use additional modules (in this case, lecture)	4	Perfect encapsulation
301959	299724	2	Warning: ask Mr. Boulic if you can use additional modules (in this case, world) Warning [A1]: main.cc is not supposed to include error.h	4	Perfect encapsulation
301994	301956	3	Warning: ask Mr. Boulic if you can use additional modules (in this case, lecture)	4	Perfect encapsulation
301998	297176	3	Excellent travail sur l'architecture, bravo ! Attention cependant à ne pas fuir l'orienté-objet, vu votre niveau une classe Simulation regroupant la lecture, le test des erreurs et la création des instances aurait été parfait ! Warning : il est important de justifier en commentaire le besoin que vous avez eu de créer les modules lecture et detectionErreur, bien qu'on devine aisément qu'ils sont sous-modules de Simulation	4	L'encapsulation est correcte et bien comprise. Il est toutefois inutile de garder une variable à jour censée contenir le nombre de Player ou Ball en jeu : un simple appel à vector.size() vous retourne cette info et vous évite de potentielles erreurs dues à un oubli de maintien d'une variable compteuse. Warning : il faut corriger vos includes : on inclut QUE dans le .h, et le .cc contient uniquement son propre .h en include. Ball.cc n'a donc en include que "ball.h"
302017	287459	3	Excellent respect de l'architecture ! RAS	4	Très bonne modularisation et encapsulation. Les modules player,ball et map sont peut-être un peu modestes par rapport à simulation mais c'est un choix toléré.
302031	301750	3	Parfait	4	Parfait
302076	270354	2	{A1}: only one fonction of simulation module must be called by main. Warning: ask permission to Mr. Boulic before using additional modules (in this case, renaming map as obstacles)	1,5	[C1]: public attricute ERROR_MODE_KEYWORD in class simulation (in fact, you don't need to have this attribute, comparing argv[2] with just the string "Error" is not a mistake, or as an alternative you can move his definition in the main. Ideally, if you rellay want to keep it, make it const std::string). [C2]: all your constructors have their initialisation list in the .h

302276	301411	3		4	Warning: constructors must be implemented in the source file (convention) There seem to be quite a lot of static variables : using a class for a simulation context, the static variables of this module would simply become attributes, and the lists of instances of other classes could be stored at this higher level too, within this simulation class (instead of within the modules themselves)
302289	301766	3	Warning : dans le cas d'ajout de modules, bien penser à en justifier la raison dans le fichier .h à l'aide de commentaires. Si des modules tels qu'obstacle me paraissent justifiés, d'autres me paraissent moins nécessaires (sans être indésirables non plus), tels que leture ou tests qui auraient leur place au sein de simulation ou des modules correspondant aux divers éléments du jeu, module qui pour l'instant est un peu vide. Dans tous les cas, si vous ajoutez des modules => justifiez votre choix dans les commentaires des .h respectifs.	4	Bon travail pour la modularisation et l'externalisation
302327	302584	2	Très bon respect de l'architecture ! Cependant, le module tools DOIT rester de bas niveau et ne donc contenir aucune info ou référence sur le projet. Il est donc formellement interdit de lui inclure define.h et encore moins d'utiliser SIDE et COEF_MARGE_JEU. Corrigez cela pour le prochain rendu, ce n'est pas une grosse erreur ;)	4	Bon travail ! La modularisation et encapsulation des classes sont très bien.
302360	295890	3	Ok.	3,5	[C2] tool.h. Les constructeurs doivent être définis dans le .cc
302551	301001	2	[A2]simulation: Bonne implémentation mais votre module simulation fait la quasi-totaltié du module à sa place: déléguer correctment	4	Parfait
302565	301949	3	Ok.	4	Ok. Pas besoin de déclarer le destructeur si vous ne le redéfinissez pas. Pour éviter les interfaces trop chargés choisissez bien ce à quoi l'utilisateur a accès. Certaines méthodes d'aides peuvent être définies dans le .cc sans les déclarer dans le .h.

302583	282836	2	You should not include all files in all the files ! (Especially if you don't use them ...)	4	Good
302584	302327	2	Très bon respect de l'architecture ! Cependant, le module tools DOIT rester de bas niveau et ne donc contenir aucune info ou référence sur le projet. Il est donc formellement interdit de lui inclure define.h et encore moins d'utiliser SIDE et COEF_MARGE_JEU. Corrigez cela pour le prochain rendu, ce n'est pas une grosse erreur ;)	4	Bon travail ! La modularisation et encapsulation des classes sont très bien.
302659	296169	3	Excellent respect de l'architecture du module projet et du Modèle. Warning : Vous avez oublié de changer le nom de classe "Obstacle" dans tools. J'ai pris en compte le commentaire pour cette fois.	4	Parfait ! Pensez peut-être à utiliser la puissance des méthodes (accès aux attributs private) et de leur encapsulation lorsque c'est possible plutot que de définir des fonctions hors classe
302674	300559	3	Warning: ask Mr. Boulic if you can use additional modules (in this case, setup)	2	[C1]: ball.cc and player.cc have global attribute rayon
302703	300253	3	Good job!	4	Perfect
302747	298011	3		4	Be careful with static variables in modules. Some possibilites to use them less: When you declare a list of As in the module where you define A, you could also handle this list in a "higher" module (e.g. list of balls in simulation) For your static variables in simulation, you may want to encompass a whole context of the simulation in a class, and those variables could become attributes.
302759	301595	1	[A2]Vous ne déléguez pas assez aux modules ball et player. Vous les utilisez comme simples espaces de stockage. [A3]Le module tool travaille sur des joueurs et des balles alors qu'elles doivent lui être opaques.	2,5	[C2]Constructeur obstacle, Constucteur Balle, Constructeur Player
302896	300558	3	Les principes de l'architecture sont compris mais quelle machine de guerre que projet ! Il est inutile d'aller aussi loin, un simple argv[1] == "Error" est bien suffisant, avec à la rigueur quelques checks pour s'assurer qu'un input faux ne peux faire planter le programme.	4	Le principe d'encapsulation est bien respecté. Warning : vous faites des includes dans les .cc, ils sont censé se trouver dans les .h, le .cc ne contenant que l'include vers le .h correspondant. Warning : un prototype rendu visible dans le .h ne doit pas se trouver également au sommet du .cc.

302909	301633	3	Ok. Vu que vous utilisez une hierarchie de classes avec Cercle les méthodes de collisions peuvent être mises directement dans la classe. Cela permettra aussi de spécialiser le comportement en cas de collisions. Ce qui, pour l'instant n'est pas beaucoup exploité par votre code. Repensez à la relation "a un" et "est un".	4	Ok.
303011	288252	3	L'architecture est bien respectée	4	Malgré les bémols qui suivent, les grands principes d'encapsulation sont bien respectés. Warning: vous définissez vos vecteurs comme globaux au module qui les contient. Il semble plus logique de placer ces variables statique dans la partie private de votre classe pour en faire des attributs bien encapsulés. Warning : projet.cc contient des includes inutiles. Warning : toutes vos classes sont définies avec des fonctions à part, dans un style purement séquentiel. Or ces fonctions devraient être des méthodes des classes, c'est une violation du principe d'Orienté-objet. Vous devez corriger ça pour le futur.
303028	296372	3	Très bien	4	Très bien. Est-il nécessaire de déclarer des constantes dans projet.cc alors que celles-ci ne sont utilisées qu'une seule fois chacune, et ne sont pas amenées à évoluer ?
303039	283652	3	Good	2,5	[C2] tools.h constructor, player.h, map.h headers file should only contains method prototype (or exceptionally definition on the same line as the prototype)
303153	301836	3	warning : justifiez l'utilisation des .h autres que ball,player et map dans simulation.h. Tentez de déleguer plus aux autres classes et mettez moins de code dans simulation.cc.	4	Ok
305538	270354	2	{A1}: only one fonction of simulation module must be called by main. Warning: ask permission to Mr. Boulic before using additional modules (in this case, renaming map as obstacles)	1,5	[C1]: public attricute ERROR_MODE_KEYWORD in class simulation (in fact, you don't need to have this attribute, comparing argv[2] with just the string "Error" is not a mistake, or as an alternative you can move his definition in the main. Ideally, if you rellay want to keep it, make it const std::string). [C2]: all your constructors have their initialisation list in the .h