Sciper	**Architecture (max 2pts)**	Architecture violation comment	**CLASS (max 3pts)**	Class violation comment	**Coding Style (max 5pts)**	violation list	violation comment	General comment	**TOTAL (max 10pts)**
302064	**2**	OK	**1**	[C1] simulation.cc:12 variable globale, pour la limiter à la portée du module il faut la mettre dans l'unnamed namespace ou inclure le mot-clé static [C2] robot.h:33-39, méthode non externalisée WARNING:using namespace std ne devrait pas apparaitre les .h	**3**	[L1] particule.h:13,16 ; robot.h:14,16,23,32,... ; partout [P2] simulation.cc:35-118 Fonction de 84 lignes	private & public ne doivent pas être indentés. WARNING: inconsistance sur les indentations: accolade ouvrante sur la même ligne sauf pour les fonctions. À corriger, il faut que ce soit partout pareil.	Bon travail. Faites néanmoins attention au style et à l'encapsulation. Par ailleurs, vos classes robots ne comportent aucune méthode en dehors du constructeur, ce qui est un peu dommage du point de vue de l'orienté objet.	**6**
326025	**1**	[A3] "constantes.h" inclus dans shape, à enlever	**3**	OK	**4**	[L2]particule.cc57,67,72,86	WARNING - Ne pas indenter les termes "public/private"; plusieurs lignes dépassent les 87 charactères	Bon travail, présentation génerale du code à revoir	**8**
328364	**2**	ok	**3**	ok	**4**	[L2]simulation.cc:45,46,62,72,73,...	vous avez complètement ignoré la limite de longueur de ligne ; vous avez le droit de passer à la ligne et d'aligner la suite d'une instruction	la fonction de lecture est assez opaque pour tenir dans les 40 lignes ; plus d'abstraction aurait été bienvenue. Attention aux performances à cause des nombreux passages de vector par valeur. Il faudra déterminer où les vector de robots et autres seront mémorisés car actuellement il s'agit de variables locales.	**9**
328433	**2**	ok	**3**	ok	**5**		ok	Bien	**10**
329207	**1**	[A3] Le module shape ne peut inclure aucun module du niveau supérieur, il ne peut donc pas inclure constantes.h	**3**	WARNING: il ne faut pas mettre de "using namespace" dans les .h WARNING: particule.h:17 & robot.h:18,35,... la définition de méthodes dans le .h est en effet autorisée uniquement pour les getters et les constructeurs, mais seulement si elle tient sur la même ligne. Ne passez pas donc pas à la ligne.	**5**	[L1] shape.cc:74,78,91,94 [L2] particule.cc:15, robot.cc:18 [021] particule.h, shape.h [E13] simulation.cc, simulation.h	WARNING: Indentation inconsistante (un cran trop loin) Pas pénalisé car moins de 4 fois WARNING: Dépassement de 87 caractères Pas pénalisé car moins de 4 fois WARNING: format des header guards: MON_SOURCE_H pour mon_source.h WARNING: le nom d'une classe doit débuter par une Majuscule	En dehors de quelques warnings de style et du fait que votre module shape inclut des modules du niveau supérieur, votre rendu est très bon, beau travail.	**9**

329309	**2**	ok	**3**	ok	**5**		ok	Bien	**10**
329942	**2**	ok	**3**	ok	**5**		ok	Code impeccable. Bravo!	**10**
330515	**0**	[A1] Aucun module projet, [A2] Aucun module du modèle, [A3] tout le programme est visiblement implémenté dans l'unique module shape	**0**	[C2] Aucune méthode n'est externalisée WARNING: votre code n'a aucune hiérarchie, toutes les classes sont définies dans le même module shape	**4**	[L2] shape.cc:513,550,570, 571,584,...	Ne pas dépasser la limite de 87 caractères par ligne.	Vous respectez généralement les conventions de style en dehors de la limite de 87 caractères. Cependant, vous avez complètement ignoré les principes de l'orienté-objet (abstraction, encapsulation) en mettant tout votre code dans shape.cc . Ceci est à absolument corriger pour le prochain rendu.	**4**
330686	**2**	OK	**1**	[C1] simulation.cc:12 variable globale, pour la limiter à la portée du module il faut la mettre dans l'unnamed namespace ou inclure le mot-clé static [C2] robot.h:33-39, méthode non externalisée WARNING:using namespace std ne devrait pas apparaitre les .h	**3**	[L1] particule.h:13,16 ; robot.h:14,16,23,32,... ; partout [P2] simulation.cc:35-118 Fonction de 84 lignes	private & public ne doivent pas être indentés. WARNING: inconsistance sur les indentations: accolade ouvrante sur la même ligne sauf pour les fonctions. À corriger, il faut que ce soit partout pareil.	Bon travail. Faites néanmoins attention au style et à l'encapsulation. Par ailleurs, vos classes robots ne comportent aucune méthode en dehors du constructeur, ce qui est un peu dommage du point de vue de l'orienté objet.	**6**
333256	**1**	[A1] shape.h inclus dans projet.cc [A3] constante.h inclus dans shape.cc	**3**	WARNING: il devrait y avoir 3 types de robots (3 classes)	**5**	ok	WARNING: ne pas indenter private, public	C'est bien, continuez comme ça, vous êtes sur la bonne voie pour la suite du projet. Revoyez un peu les classes avant d'entamer la suite.	**9**
339452	**1**	[A3] "constantes.h" inclus dans shape, à enlever	**3**	OK	**4**	[L2]particule.cc57,67,72,86	WARNING - Ne pas indenter les termes "public/private"; plusieurs lignes dépassent les 87 charactères	Bon travail, présentation génerale du code à revoir	**8**
339655	**1**	[A2]"simulation.h" inclus dans "particule.cc"	**2**	[C1]Variable globale "spatial" dans simulation.cc, à enlever	**4**	[L2]particule.cc31,33,44,51...	Plusieurs lignes qui dépassent les 87 charactères au fil du code (particule.cc, shape.cc, robot.cc)	Bonne décomposition, faire attention à l'architecture de projet	**7**

339837	**1**	[A3]: Shape depends on constantes.h and it contains comments about robots and particules Suggestions: 1. In the project, there are precompiled header files even if your makefile doesn't target any header files, you can remove them, 2. You don't need a project.h 3. Useless header includes in multiple files 4. In simulation.h, the header guard does not include all the code and it has a different name from the file	**1**	[C2]: robot.h, particule.h: the complete definition of both classes is in the header file [C2]: shape.h NONCONSTANT global variable eps (and outside a namespace) WARNING: 1. robot.h, particules.h: global static vectors in a header file, they should be moved in .cc	**4**	[L1] particule.h: class methods are not indented , robot.cc: incosistent bracing and spacing of the braces, shape.cc same ...	Warnings: 1. Try to find better names for variables: nbNs_nbRs_nbupdate, sparticules1... 2. Inconsistent naming style for variables/methods: camelCase vs snake_case 3. General inconsistent spacing, use a code formatter!	The code is not very readeable nor understandable. Try to simplify it and organize it: 1. Separate the robot class in multiple subclasses, in order to simplify the class robot 2. Choose a naming style and stick to it 3. Better variable/method names 4. Try to simplify the overall structure: different vector particules with similar name modify in different files (for example create a class Simulation that owns all these vectors)... 5. Use vectors of pointers to classes to classes and reference in order to not copy everytime the vectors	**6**
339870	**2**	[A1] ATTENTION: faites passer l'appel du message de succès dans le modèle (simulation), il ne peut pas être contenu dans projet.cc. Ne pas importer message.h.	**3**	OK	**5**	OK	Attention aux inconsistances (std:: & std ::, nom de fonctions en majuscules dans shape, etc)	Excellent rendu! Corrigez simplement l'utilisation de message dans projet, et faites un petit effort pour la consistance du style.	**10**
339959	**1**	[A1]: Le module projet ne gère que les arguments de la ligne de commande (argc et argv). Seule une méthode de simulation doit être appelée. (check_superpos() doit être appelée dans une méthode de simulation); WARNING: inclusion inutile de shape.h dans simulation.h ainsi que de particule.h et message.h dans robot.h; inutile de refaire des includes qui sont déjà dans le header du module (ex: particule, robot) ; Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	WARNING: Utilisation excessive des setters. Une bien meilleure méthode serait d'utiliser les constructeurs. Les setters peuvent être utilisés dans la définition des constructeurs pour vérifier la validité des paramètres transmis.	**4**	[P2] simulation.cc: 44-156	La fonction decodage_ligne dépasse le maximum de 80 lignes autorisé pour une fonction; WARNING: [L2] dépassement de la limite des 87 caractères à deux reprises (simulation.cc: 99,146)	Code très lisible. Attention néanmoins à bien respecter le principe d'encapsulation en utilisant des constructeurs et en limitant au maximum l'utilisation de setters	**8**
339964	**2**	ok	**3**	ok	**4**	[L2] simulation.h 33, simulation.cc, robot.h etc		C'est globalement un bon travail, continuez comme ça	**9**
340062	**2**	Ok	**3**	Ok	**5**		Ok	Excellent rendu, quelques remarques: les noms des fonctions ne sont pas très descriptifs et vos aérations ne sont pas consistentes	**10**

340435	**2**	Top	**3**	Suggestions: 1. shape.h: epsil_zero should be in a named namespace	**5**	Top	Warnings: 1. Inconsistent spacing: use a code formatter! 2. Don't chain statement in one single line 3. Inconsistent naming style for variables/methods: camelCase or snake_case 4. Sometimes the names of variables/methods are not clear enough 5. Switch cases need brackets only if temporary variables are used	The code is not very readeable (especially shape) and sometimes convoluted due to the overall architecture chosen: 1. In C++17, instead of using your own defined type vecteur, you can use std::pair or std::tuple 2. Divide long functions (for example decodage_ligne_particule, decodage_ligne_robot_n) in multiple functions with different goals 3. In simulation.cc, the function decodage_ligne is a little bit convoluted as it doesn't have access to data already read, try to create a class Simulation with space robot and the others robots as attributes, so you don't have go back and forth from each modules to access the information 4. The module shape is barely readable, add spaces and divide the functions 5. Divide the class Robot in multiple different subclasses to simplify it 6. Pass struct and classes by reference and use vector of pointers (better yet unique_ptr) to classes and structs in order to avoid useless copies	**10**
340497	**1**	warning: inutile d'avoir un constantes.cpp [A1] Le robot spatial ne devrait pas être initialisé dans projet.cc, c'est le job de simulation de faire cela. La seule connexion que doit avoir projet au modèle est une fonction/méthode de simulation.	**3**	OK	**5**	OK	OK	Rendu impeccable en dehors de votre initialisation de Spatial dans le module projet. Excellent travail!	**9**
340863	**2**	OK	**1**	[C2]Définition des méthodes à externaliser; ne pas définir de méthodes dans l'interface des modules. C'est le cas pour robot.h, particule.h	**5**		WARNING - Ne pas indenter les termes "public/private"	Très bon travail globalement, réparer les 2 interfaces de module spécifiées pour le prochain rendu	**8**

341002	**2**	ok	**3**	ok	**4**	[L1]robot.cc:25,29-31,59-60, 70-71, 75;	warning: les noms de variable, de champ ou d'attribut ne peuvent pas être entièrement en majuscule (ex: C1, C2, L1...); améliorer les alignements (ex: shape.cc:9,16,25,27); curieux commentaire robot.cc:17, la liste d'initialisation doit êtr eindentée ou alignée, le contenu du namespace de robot.cc doit être indenté ; utilisz le même style d'accolade que pour les blocs; [L1] vous avez plusieurs styles d'accolades fermantes ; ne pas fermer plusieurs accolades sur la même ligne (75).	Code bien structuré mais respectez l'organisation des fichiers (ex: robot.cc le namespace non-nommé et les typedef sont avant la définition des méthodes); certaines variable locale devront être mémorisées à plus long terme (robot.cc:91). pour le rendu2 il faudra prévoir la ré-initialisation des variables static de lecture de fichier (simulation.cc).	**9**
341055	**2**	ok	**3**	WARNING: Simulation doit être une classe, les nombreux unamed namespace ne font que compliquer le programme	**5**	[L2]robot.cc144,163		Code dur à suivre entre les namespace et les classes avec le meme nom.	**10**
341138	**2**	ok	**3**	ok	**5**	ok		Pourquoi passer l'instance de simulation a simulation ? Bonne décomposition autrement	**10**
341237	**1**	warning: inutile d'avoir un constantes.cpp [A1] Le robot spatial ne devrait pas être initialisé dans projet.cc, c'est le job de simulation de faire cela. La seule connexion que doit avoir projet au modèle est une fonction/méthode de simulation.	**3**	OK	**5**	OK	OK	Rendu impeccable en dehors de votre initialisation de Spatial dans le module projet. Excellent travail!	**9**
341489	**2**	OK	**3**	OK	**5**		Warning - [L2]spatial.cpp31, faire attention aux lignes trop longues	Très bon travail, code impeccable. Warning en ce qui concerne les modules en plus, il vaut mieux tout mettre dans un module Robot plutot que de séparer chaque sous-classe dans un module à part. Il vaut mieux prendre l'autorisation du Prof. Boulic avant de créer des modules en plus.	**10**
341508	**2**	ok	**3**	WARNING: Simulation doit être une classe	**5**	ok	Pas d'espace entre std et ::	Attention a bien utiliser le code fourni (success). Simulation doit être dans une classe, autrement très bien	**10**

341509	**2**	Wanring: faire les comparaisons entre cercle et carré dans le module shape	**3**	ok	**5**	ok	Utiliser des noms clair (verify_superpos_particle_robot ne verifie rien)	Les comparaisons entre cercle et carres devraient se faire dans le module shape, ce qui simplifierai le code étant données que les comparaisons sont actuellement faitent dans les modules robots et partcules	**10**
341530	**2**	OK	**2**	[C2]simulation.h; l'interface d'une classe ne doit pas contenir des définitions de Enum	**5**		OK	Très bon travail globalement	**9**
341668	**1**	[A2]"simulation.h" inclus dans "particule.cc"	**2**	[C1]Variable globale "spatial" dans simulation.cc, à enlever	**4**	[L2]particule.cc31,33,44,51...	Plusieurs lignes qui dépassent les 87 charactères au fil du code (particule.cc, shape.cc, robot.cc)	Bonne décomposition, faire attention à l'architecture de projet	**7**
341672	**2**	WARNING message.h dans projet.cc	**3**	ok	**5**	WARNING: [L1] surindentation: shape.h 15, 28, 26		Bravo, bonne continuation pour la suite du projet	**10**
341744	**2**	ok	**3**	WARNING:using namespace std ne devrait pas apparaitre les .h	**4**	[L1] particule.h: 15,21; robot.h: 14,19,27,32,39,47,55,61	Il ne faut pas indenter les mots-clés public, private et protected	Très bien	**9**
341898	**2**	Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	ok	**4**	[L1] robot.h: 20,23,37,41,46,51,58,62; particule.h:16,24	Il ne faut pas indenter les mots-clés public, private et protected; WARNING: dépassement de 87 caractères par ligne: robot.cc: 92	Globalement, code très lisible. Il est néanmoins inutile de définir des getters pour des structures. Il est aussi inutile de répéter les includes qui sont dans les header files dans les fichiers .cc.	**9**

341903	**2**	Top Warnings: 1. Do not include useless headers (e.g project.cc) 2. If the prototype of a function is already defined in the header files, you don't need to add its prototype also in the .cc	**2**	[C1] simulation.h: global variables should be declared in a unnamed namespace in the .cc not in the header Suggestions: 1. shape.h: epsil_zero should be in a named namespace	**4**	[L1] simulation.cc: all for loop should indented [L1] simulation.cc: only one inline stament and do not mixed between inline and bracketed ifs	Warnings: 1. Inconsistent spacing, use a code formatter! 2. Inconsistent naming style for variables/methods: camelCase or snake_case 3. Add more space to your methods 4. Do not mix english and french for the same method's name, it leads to strange and not understandable names 5. Do not use 0/1 for boolean, either use true / false or enum (in order to ignore epsilon_zero) 6. Use for(auto s:ss) whenever possible as it simplify the code	The code is readeable and understanble, but sometimes your methods are too compacted, therefore even if your function are smaller, do not sacrifice readability for space: write complete if statement (e.g shape eps?...), write longer and better names... 1. The constant PI is already defined in the library cmath 2. In simulation, instead of using global vectors, you could create a Simulation class that owns all the vectors, thereby simplify the overall structure of your code 3. The function decode is very convoluted and not very readeable, try to simplify it and divide it in multiple smaller function 4. Sometimes your variables could have better names	**8**
341942	**2**	OK	**3**	OK	**5**		OK	Rien à redire, vous avez bien pensé à votre implémentation et votre code utilise bien les concepts vu en cours. Continuez ainsi pour le rendu 2.	**10**
341949	**1**	[A3] inclusion de constantes.h dans shape Warning: Eviter d'inclure des modules que vous n'utilisez pas (ex: main.cc inclusion de particule.cc et robot.cc) ou d'inclure des modules dans le header qui pourraient que figurer dans le .cc (ex: particule.h inclusion de string et constantes)	**2**	[C1] robot.h:24 la hitbox robot devrait être protected	**3**	[L2] shape.h: 26,33,53,45,etc... [P2] simulation.cc:decodage_ligne	Le module shape ne respecte pas les conventions d'indentations. La fonction decodage_ligne est trop grande (max:80) penser à la refactoriser pour le prochain rendu.	Dans l'ensemble code est bien écrit et clair. C'est vraiment dommage pour les quelques fautes d'inattention qui vous enlève quelques points. Sinon super travail.	**6**
341970	**2**	ok	**3**	ok	**5**		ok	Code impeccable. Bravo!	**10**
342101	**2**	OK	**3**	OK	**4**	[L2] simulation.cc:22,107; robot.cc:33,49,52,53,59; particule.cc:40-41,...	Dépassement des 87 caractères par ligne.	Excellent rendu, beau travail. Attention aux dépassement de 87 caractères, cela inclut également les commentaires.	**9**
342224	**1**	[A3] shape n'a pas connaissance des robots ni particules, que des formes primitives	**3**	OK	**5**	[L2] robot.cc:43,76,169;robot.h:85	Petits oublis dans le wrapping.	Très bon code, bien structuré et utilisant intelligemment les principes de la programmation orientée objet. Il faut juste corriger la nomenclature dans votre module shape. Continuez ainsi pour le rendu 2.	**9**

342283	**2**	OK	**3**	OK	**4**	[L1] Toutes impélentations de constructeurs [P2]simulation.cc:94	Problèmes de double indentation pour les constructeurs.	Le projet est bon, vous pourriez toutefois placer vos tests de collision plus localement. Continuez vos effort pour le rendu 2.	**9**
342391	**2**	Warning: inclusion inutile de shape.h dans simulation.h; inclusion inutile de cmath et message.h dans robot.h; inclusion inutile de message.h et constante.h dans particule.h; À ENLEVER ET À METTRE DANS LES .cc correspondants	**3**	ok	**4**	[L2] robot.h: 16-18; robot.cc: 5-10-16-55; simulation.cc :136-141.....	Plusieurs lignes dépassent la limite acceptée de 87 caractères (mauvais réalignement après les retours à la ligne)	Globalement, votre code est assez clair et respecte l'architecture imposée.	**9**
342575	**2**	WARNING: include inutile de message.h dans robot.h et particule.h	**3**	ok	**5**	[L2] : robot.h: 64; robot.cc: 25,95	WARNING: Dépassement de 87 caractères Pas pénalisé car moins de 4 fois	Globalement, votre code est assez clair et respecte l'architecture imposée. Bon travail	**10**
342906	**2**	Wanring: faire les comparaisons entre cercle et carré dans le module shape	**3**	ok	**5**	ok	Utiliser des noms clair (verify_superpos_particle_robot ne verifie rien)	Les comparaisons entre cercle et carres devraient se faire dans le module shape, ce qui simplifierai le code étant données que les comparaisons sont actuellement faitent dans les modules robots et partcules	**10**
342907	**1**	[A3]: Shape depends on constantes.h and it contains comments about robots and particules Suggestions: 1. In the project, there are precompiled header files even if your makefile doesn't target any header files, you can remove them, 2. You don't need a project.h 3. Useless header includes in multiple files 4. In simulation.h, the header guard does not include all the code and it has a different name from the file	**1**	[C2]: robot.h, particule.h: the complete definition of both classes is in the header file [C2]: shape.h NONCONSTANT global variable eps (and outside a namespace) WARNING: 1. robot.h, particules.h: global static vectors in a header file, they should be moved in .cc	**4**	[L1] particule.h: class methods are not indented , robot.cc: incosistent bracing and spacing of the braces, shape.cc same ...	Warnings: 1. Try to find better names for variables: nbNs_nbRs_nbupdate, sparticules1... 2. Inconsistent naming style for variables/methods: camelCase vs snake_case 3. General inconsistent spacing, use a code formatter!	The code is not very readeable nor understandable. Try to simplify it and organize it: 1. Separate the robot class in multiple subclasses, in order to simplify the class robot 2. Choose a naming style and stick to it 3. Better variable/method names 4. Try to simplify the overall structure: different vector particules with similar name modify in different files (for example create a class Simulation that owns all these vectors)... 5. Use vectors of pointers to classes to classes and reference in order to not copy everytime the vectors	**6**
344310	**2**	Top	**3**	Top	**5**	Top	Top	The code is excellently written and easy to understand Suggestions: 1. In order to avoid useless copies, use vectors of unique_ptr for the robots	**10**

344321	**2**	Top	**3**	Top	**5**	Top	Top Suggestions: 1. Incosistent spacing, use a code formatter! 2. In order to avoid copy struct and vector use const reference and vectors of pointers	The code is readable and overall easy to understand, goob job! Suggestions: 1. Try centralize the ownership of the lists, for example by adding them to the class Simulation. 2. Use pointers and reference in order to avoid copying everytime structs, classes and vectors. 3. Divide the class Robot in multiple classes in order to simplify it 4. Remove .o from the project before uploading it	**10**
344325	**2**	ok	**3**	ok	**4**	[P2]simulation.cc92	Surindentation dans les classes: public et private ne doivent pas être indentés		**9**
344416	**2**	Ok Warnings: 1. Very complicated definition of the instance of Simulation, simply define it in project.cc	**2**	[C1]: simulation.cc: the class Simulation has as static attribute itself, therefore by using the method getistance that return the static attribute by reference, it is possible to modify it outside the class --> breaks the encapsulation and too complicated... simply add it as variable in project.cc Suggestions: 1. shape.h: epsil_zero should be in a named namespace	**4**	[L1] robot.h: do not indent private, public [L1] shape.cc: mixed indent spaces [L1] simulation.cc: mixed indentation style for decodage_ligne	Warnings: 1. Incosistent naming case for variables/methods: camelCase or snake_case 2. All classes must start with an uppercase 3. Overall incosistent spacing: use a code formatter! 4. Do not use typedef to define a type that you will use one time, simply use a good name for the variable 5. Try to give better names to methods/variables...	The code is not very readable and sometimes convoluted. Try to give better names to variables and methods, divide big functions into smaller functions with each a specific task: 1. In project.cc you don't need to check whether or not the first argument is equal to "./projet", as it is the name of the executable 2. In the module shape, the function distance is set as static without any apparent reason 3. As already written, fix the Simulation::gestinstance 4. Simulation::decodage_ligne is a little bit convoluted as there are: multiple different variables with similar names, functions called test but don't return nothing and instead directly modify the vectors, incosistent spacing...	**8**
344529	**2**	OK	**3**	OK	**5**	[L1]shape.cc, particule.cc31	Mauvaises indentations	Très bon travail	**10**

344936	**2**	Top	**3**	Suggestions: 1. shape.h: epsil_zero should be in a named namespace	**5**	Top	Warnings: 1. Inconsistent spacing: use a code formatter! 2. Don't chain statement in one single line 3. Inconsistent naming style for variables/methods: camelCase or snake_case 4. Sometimes the names of variables/methods are not clear enough 5. Switch cases need brackets only if temporary variables are used	The code is not very readeable (especially shape) and sometimes convoluted due to the overall architecture chosen: 1. In C++17, instead of using your own defined type vecteur, you can use std::pair or std::tuple 2. Divide long functions (for example decodage_ligne_particule, decodage_ligne_robot_n) in multiple functions with different goals 3. In simulation.cc, the function decodage_ligne is a little bit convoluted as it doesn't have access to data already read, try to create a class Simulation with space robot and the others robots as attributes, so you don't have go back and forth from each modules to access the information 4. The module shape is barely readable, add spaces and divide the functions 5. Divide the class Robot in multiple different subclasses to simplify it 6. Pass struct and classes by reference and use vector of pointers (better yet unique_ptr) to classes and structs in order to avoid useless copies	**10**
345007	**2**	ok, module suppémentaire pertinent.	**0**	[C1]particule et robot. La modification des attributs de simulation (les liste de particules et robot) se fait par reference avec des fonctions d`autre modules(...init), ce qui viole le principe d`encapsulation.	**5**		Beau code, agréable à lire et respectant bien les conventions	Bravo pour l`organisation de votre code et l`excellent respect des conventions. Cependant, il faudra modifier pour le prochain rendu la maniere de modifier les attributs de simulation, en utilisant par exemple des setters	**7**

345130	**0**	[A1] Aucun module projet, [A2] Aucun module du modèle, [A3] tout le programme est visiblement implémenté dans l'unique module shape	**0**	[C2] Aucune méthode n'est externalisée WARNING: votre code n'a aucune hiérarchie, toutes les classes sont définies dans le même module shape	**4**	[L2] shape.cc:513,550,570, 571,584,...	Ne pas dépasser la limite de 87 caractères par ligne.	Vous respectez généralement les conventions de style en dehors de la limite de 87 caractères. Cependant, vous avez complètement ignoré les principes de l'orienté-objet (abstraction, encapsulation) en mettant tout votre code dans shape.cc . Ceci est à absolument corriger pour le prochain rendu.	**4**
345259	**1**	[A3] shape doit être INDEPENDANT de son utilisation dans votre projet	**3**	Ok	**3**	[L1] Tout code [L2] Tout code	Vous n'avez pas fait attention au wrapping. Il ne doit pas y avoir d'indentation au niveau des mots clés public et private. Des erreurs d'indentation dans simulation.cc aussi	Votre code est assez efficace donc on voit qu'il y a eu de la réflexion mais vous avez du faire quelques erreurs à cette étape. Attention à bien respecter les consignes, mais cela peut s'arranger assez facilement pour le rendu 2. Penser aussi à placer vos tests de collisions plus localement.	**7**
345308	**2**	ok	**3**	WARNING: Simulation doit être une classe	**5**	ok	Pas d'espace entre std et ::	Attention a bien utiliser le code fourni (success). Simulation doit être dans une classe, autrement très bien	**10**
345310	**2**	Warning - message.h ne peut pas etre inclus dans projet.cc, à changer pour le prochain rendu	**2**	[C2]Définition des méthodes à externaliser; ne pas définir de méthodes dans l'interface des modules. C'est le cas pour robot.h; il faut definir les constructeur à l'interieur du fichier .cc	**5**		OK	Très bon travail globalement	**9**
345337	**2**	ok	**3**	ok	**4**	[L1] missing indentation in namespace shape.cc,, shape.h	Il y a souvent une surindentation (module robot, particule.cc), WARNING: vous utilisez une fonction Lecture dsans projet.cc qui n est pas defini et vous n'utilisez pas la fonction lecture de votre module simulation	Vous avez pris un bon départ, continuez comme ça.	**9**

345473	**2**	Top Warnings: 1. Do not include useless headers (e.g project.cc) 2. If the prototype of a function is already defined in the header files, you don't need to add its prototype also in the .cc	**2**	[C1] simulation.h: global variables should be declared in a unnamed namespace in the .cc not in the header Suggestions: 1. shape.h: epsil_zero should be in a named namespace	**4**	[L1] simulation.cc: all for loop should indented [L1] simulation.cc: only one inline stament and do not mixed between inline and bracketed ifs	Warnings: 1. Inconsistent spacing, use a code formatter! 2. Inconsistent naming style for variables/methods: camelCase or snake_case 3. Add more space to your methods 4. Do not mix english and french for the same method's name, it leads to strange and not understandable names 5. Do not use 0/1 for boolean, either use true / false or enum (in order to ignore epsilon_zero) 6. Use for(auto s:ss) whenever possible as it simplify the code	The code is readeable and understanble, but sometimes your methods are too compacted, therefore even if your function are smaller, do not sacrifice readability for space: write complete if statement (e.g shape eps?...), write longer and better names... 1. The constant PI is already defined in the library cmath 2. In simulation, instead of using global vectors, you could create a Simulation class that owns all the vectors, thereby simplify the overall structure of your code 3. The function decode is very convoluted and not very readeable, try to simplify it and divide it in multiple smaller function 4. Sometimes your variables could have better names	**8**
345683	**2**	Warning - message.h ne peut pas etre inclus dans projet.cc, à changer pour le prochain rendu	**2**	[C2]Définition des méthodes à externaliser; ne pas définir de méthodes dans l'interface des modules. C'est le cas pour robot.h; il faut definir les constructeur à l'interieur du fichier .cc	**5**		OK	Très bon travail globalement	**9**
345862	**2**	WARNING message.h dans projet.cc	**3**	ok	**5**	WARNING: [L1] surindentation: shape.h 15, 28, 26		Bravo, bonne continuation pour la suite du projet	**10**
346154	**2**	Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	ok	**4**	[L1] robot.h: 20,23,37,41,46,51,58,62; particule.h:16,24	Il ne faut pas indenter les mots-clés public, private et protected; WARNING: dépassement de 87 caractères par ligne: robot.cc: 92	Globalement, code très lisible. Il est néanmoins inutile de définir des getters pour des structures. Il est aussi inutile de répéter les includes qui sont dans les header files dans les fichiers .cc.	**9**
346228	**2**	[A2] WARNING: demander dès maintenant à Ronan Boulic l'autorisation d'utiliser votre architecture non-conventionnelle	**3**	ok	**5**	ok	ok	Très bon rendu, mais votre architecture doit être validée par l'enseignant avant de pouvoir continuer avec celle-ci.	**10**
346230	**2**	OK	**2**	[C2]simulation.h; l'interface d'une classe ne doit pas contenir des définitions de Enum	**5**		OK	Très bon travail globalement	**9**

346255	**1**	[A1] shape.h inclus dans projet.cc [A3] constante.h inclus dans shape.cc	**3**	WARNING: il devrait y avoir 3 types de robots (3 classes)	**5**	ok	WARNING: ne pas indenter private, public	C'est bien, continuez comme ça, vous êtes sur la bonne voie pour la suite du projet. Revoyez un peu les classes avant d'entamer la suite.	**9**
346389	**2**	OK	**3**	OK	**5**		Warning - [L2]spatial.cpp31, faire attention aux lignes trop longues	Très bon travail, code impeccable. Warning en ce qui concerne les modules en plus, il vaut mieux tout mettre dans un module Robot plutot que de séparer chaque sous-classe dans un module à part. Il vaut mieux prendre l'autorisation du Prof. Boulic avant de créer des modules en plus.	**10**
346483	**2**	OK	**3**	OK	**4**	[L2] particule.h:31;robot.cc:70,81;simulation.cc:142,175,183	Petits oublis dans le wrapping.	Attention à bien penser à retirer votre code de test. A part cela le projet est parfaitement réfléchi et exécuté, continuez ainsi pour le rendu 2.	**9**
346515	**1**	[A3] shape ne doit pas dépendre de constantes, c'est un module indépendant du reste de votre projet	**3**		**4**	[L1] robots.cc:70	Quelques erreurs d'indentation et d'inconsistence qui rendent la lecture de votre code moins agréable	Très bon rendu, quelques erreurs d'indentation	**8**
346518	**2**	OK	**3**	WARNING:using namespace std ne devrait pas apparaitre dans les .h	**4**	[L1] simulation.h:13,15; particule.cc:59; partout [021] robot.h	WARNING: format des header guards: MON_SOURCE_H pour mon_source.h Ne pas indenter private & public (simulation.h), inconsistance accolade (particule.cc) Il semble que dans certains fichiers, vos premiers niveaux d'indentation ont un espacement de 3 caractères, tandis que les suivant ont un espacement de 4 caractères. Cela est probablement dû au mélanges espaces/TAB. À corriger, il faut que ça soit consistant.	Excellent rendu, corrigez seulement l'indentation inconsistance	**9**

346596	**1**	[A1]: Le module projet ne gère que les arguments de la ligne de commande (argc et argv). Seule une méthode de simulation doit être appelée. (check_superpos() doit être appelée dans une méthode de simulation); Warning: inclusion inutile de shape.h dans simulation.h ainsi que de particule.h et message.h dans robot.h; inutile de refaire des includes qui sont déjà dans le header du module (ex: particule, robot) ; Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	WARNING: Utilisation excessive des setters. Une bien meilleure méthode serait d'utiliser les constructeurs. Les setters peuvent être utilisés dans la définition des constructeurs pour vérifier la validité des paramètres transmis.	**4**	[P2] simulation.cc: 44-156	La fonction decodage_ligne dépasse le maximum de 80 lignes autorisé pour une fonction; WARNING: [L2] dépassement de la limite des 87 caractères à deux reprises (simulation.cc: 99,146)	Code très lisible. Attention néanmoins à adopter les bonnes habitudes de la programmation orienté-objet en utilisant des constructeurs et en limitant au maximum l'utilisation de setters en dehors de la classe	**8**
346973	**1**	[A3] Le module shape ne peut inclure aucun module du niveau supérieur, il ne peut donc pas inclure constantes.h	**3**	WARNING: il ne faut pas mettre de "using namespace" dans les .h WARNING: particule.h:17 & robot.h:18,35,... la définition de méthodes dans le .h est en effet autorisée uniquement pour les getters et les constructeurs, mais seulement si elle tient sur la même ligne. Ne passez pas donc pas à la ligne.	**5**	[L1] shape.cc:74,78,91,94 [L2] particule.cc:15, robot.cc:18 [021] particule.h, shape.h [E13] simulation.cc, simulation.h	WARNING: Indentation inconsistante (un cran trop loin) Pas pénalisé car moins de 4 fois WARNING: Dépassement de 87 caractères Pas pénalisé car moins de 4 fois WARNING: format des header guards: MON_SOURCE_H pour mon_source.h WARNING: le nom d'une classe doit débuter par une Majuscule	En dehors de quelques warnings de style et du fait que votre module shape inclut des modules du niveau supérieur, votre rendu est très bon, beau travail.	**9**
346992	**1**	[A3] "constantes.h" inclus dans shape, à enlever; Warning - particule ne doit pas hériter de shape, revoir la donnée sur le role de shape	**1**	[C2]Définition des méthodes à externaliser; ne pas définir de méthodes dans l'interface des modules. C'est le cas pour robot.h, shape.h	**4**	[L2]robot.cc110,129,135,141,147	Faire attention aux lignes trop longues.	Bon travail, présentation du code à revoir	**6**
347058	**1**	[A3] "constantes.h" inclus dans shape, à enlever; Warning - particule ne doit pas hériter de shape, revoir la donnée sur le role de shape	**1**	[C2]Définition des méthodes à externaliser; ne pas définir de méthodes dans l'interface des modules. C'est le cas pour robot.h, shape.h	**4**	[L2]robot.cc110,129,135,141,147	Faire attention aux lignes trop longues.	Bon travail, présentation du code à revoir	**6**
347115	**2**	OK	**3**	OK	**5**		OK	Dans l'ensemble le code est bon. Pensez à utiliser des constructeurs pour gagner en lisibilté (pour éviter de faire plein d'appels à des setter). Aussi dans vos fonctions de superposition vous pourriez directement passer la hitbox de vos entités ce qui vous éviterait de devoir recréer une shape avant chaque test de collision. Bon travail!	**10**

347209	**1**	[A3] Le module shape ne peut inclure aucun module du niveau supérieur, il ne peut donc pas inclure constantes.h	**3**	OK	**3**	[L1] tous les .h [P2] simulation.cc:15-117	public & private ne doivent pas être indéntés. Fonction trop longue dans simulation: 103 lignes.	Bon travail! Faites attention au style et à l'architecture. Par ailleurs, vous faites les tests de collision (par exemple dans robot.cc) en faisant passer les coordonnées en argument; pourquoi ne pas jouir des possibilités de l'orienté objet en faisant passer un objet de shape comme Cercle?	**7**
347288	**2**	OK	**3**	OK	**4**	[L2] simulation.cc:22,107; robot.cc:33,49,52,53,59; particule.cc:40-41,...	Dépassement des 87 caractères par ligne.	Excellent rendu, beau travail. Attention aux dépassement de 87 caractères, cela inclut également les commentaires.	**9**
347341	**1**	[A3] shape.h inclus message.h. Bien faire attention, shape doit rester independant de TOUT les autres modules	**0**	[C1]particule.h,particule,particules [C1]robot.h,robot,nbUpdate Attention au attribut public, ils sont interdit. A changer pour le prochain rendu	**4**	[L1]simulation.cc56,24,25,26,robot.cc59,60,...	Attention au melange de style d`indentation et d`ouverture/fermeture de crochet	1.Pas besoin de ceer un .h s`il n`est pas nécessaire. (projet.h) 2.Les variables static ne sont pas une erreur, mais attention a ne pas en abuser. Pour votre automate de lecture, dans beaucoup de cas elles pourraient être remplacée par de simple variables locales. 3.Plutôt que d`utiliser des pairs, pourquoi ne pas mettre un attribut de plus dans Robot, ou alors utiliser du polymorphisme. Beaucoup de commentaires, mais dans l`ensemble c`est vraiment du bon travail.	**5**
347363	**1**	[A3] shape ne doit pas dépendre de message, c'est un module indépendant du reste de votre projet	**3**	Ok	**5**		Ok	Excellent rendu, très bon code, le module shape ne doit pas gérer l'affichage des messages d'erreurs mais seulement la détection des erreurs	**9**
347394	**2**	Warning: inutile de refaire des includes qui sont déjà dans le header du module robot ; Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	ok	**4**	[L1]: simulation.h: 13,19; robot.h: 11,13,23,25,35,42,55,53; particule.h:10,13	Il ne faut pas indenter les mots-clés public, private et protected	Excellente décomposition, code très lisible. Très bien. Faites juste attention de bien respecter les conventions d'indentation pour les prochains rendus	**9**
347518	**1**	[A3] Le module shape ne peut inclure aucun module du niveau supérieur, il ne peut donc pas inclure constantes.h	**3**	OK	**3**	[L1] tous les .h [P2] simulation.cc:15-117	public & private ne doivent pas être indéntés. Fonction trop longue dans simulation: 103 lignes.	Bon travail! Faites attention au style et à l'architecture. Par ailleurs, vous faites les tests de collision (par exemple dans robot.cc) en faisant passer les coordonnées en argument; pourquoi ne pas jouir des possibilités de l'orienté objet en faisant passer un objet de shape comme Cercle?	**7**

348128	**2**	ok	**3**	ok	**4**	[L1]robot.cc:25,29-31,59-60, 70-71, 75;	warning: les noms de variable, de champ ou d'attribut ne peuvent pas être entièrement en majuscule (ex: C1, C2, L1...); améliorer les alignements (ex: shape.cc:9,16,25,27); curieux commentaire robot.cc:17, la liste d'initialisation doit êtr eindentée ou alignée, le contenu du namespace de robot.cc doit être indenté ; utilisz le même style d'accolade que pour les blocs; [L1] vous avez plusieurs styles d'accolades fermantes ; ne pas fermer plusieurs accolades sur la même ligne (75);	Code bien structuré mais respectez l'organisation des fichiers (ex: robot.cc le namespace non-nommé et les typedef sont avant la définition des méthodes); certaines variable locale devront être mémorisées à plus long terme (robot.cc:91). pour le rendu2 il faudra prévoir la ré-initialisation des variables static de lecture de fichier (simulation.cc).	**9**
351248	**2**	ok	**3**	ok	**4**	[L2]simulation.cc:45,46,62,72,73,...	vous avez complètement ignoré la limite de longueur de ligne ; vous avez le droit de passer à la ligne et d'aligner la suite d'une instruction	la fonction de lecture est assez opaque pour tenir dans les 40 lignes ; plus d'abstraction aurait été bienvenue. Attention aux performances à cause des nombreux passages de vector par valeur. Il faudra déterminer où les vector de robots et autres seront mémorisés car actuellement il s'agit de variables locales.	**9**
355524	**2**	OK	**3**	OK	**4**	[L1] Toutes impélentations de constructeurs [P2]simulation.cc:94	Problèmes de double indentation pour les constructeurs.	Le projet est bon, vous pourriez toutefois placer vos tests de collision plus localement. Continuez vos effort pour le rendu 2.	**9**
355529	**1**	Attention, message.h ne doit pas être importé dans project.cc ! [A3] constantes.h ne doit pas être importé dans shape	**3**	Ok	**4**	[L1] robots.cc:18,32;robot_spacial.h:14,22;shape.h:24,32;simulation.h:36	Petits problèmes d'indentation des mots privates et publics assez récurrents et inconstants, pareil au niveau des constructeurs (notamment dans robot.cc)	Attention à l'utilisation du namespace std un peu partout dans le code. La structure est bonne mais mérite peut être une petite réflexion avant de passer au rendu 2 pour harmoniser le tout (pour la hiérarchie de classe notamment). Ils y a quelques petits soucis à corriger mais vous êes sur la bonne voie.	**8**

355583	**2**	ok	**3**	WARNING:using namespace std ne devrait pas apparaitre les .h	**3**	[L1] particule.h: 7,11; particule.cc; robot.h: 12,14,18;simulation.h: 11,14 [L2]: robot.cc: 18,40,41,43; robot.h: 30,45,46	Il ne faut pas indenter les mots-clés public, private et protected;il ne peut y avoir 2 indentations (particule.cc), plusieurs dépassements de 87 caractères dans robot.cc et robot.h	Bien mais il y a des erreurs de mise en forme à corriger	**8**
355638	**1**	[A3]"constants.h" inclus dans "shape.cc", A ENLEVER	**3**	OK	**5**		WARNING - Ne pas indenter les termes "public/private"	Très bon travail globalement	**9**
355665	**2**	OK	**3**	OK	**5**		OK	Dans l'ensemble le code est bon. Pensez à utiliser des constructeurs pour gagner en lisibilté (pour éviter de faire plein d'appels à des setter). Aussi dans vos fonctions de superposition vous pourriez directement passer la hitbox de vos entités ce qui vous éviterait de devoir recréer une shape avant chaque test de collision. Bon travail!	**10**
355726	**2**	ok	**3**	ok	**5**		ok	Très bien	**10**
355728	**2**	ok	**3**	ok	**4**	[L1]shape.cc	2 espace d'indentations	Attention à l'indentation. Bon code autrement	**9**
355732	**1**	Attention, message.h ne doit pas être importé dans project.cc ! [A3] constantes.h ne doit pas être importé dans shape	**3**	Ok	**4**	[L1] robots.cc:18,32;robot_spacial.h:14,22;shape.h:24,32;simulation.h:36	Petits problèmes d'indentation des mots privates et publics assez récurrents et inconstants, pareil au niveau des constructeurs (notamment dans robot.cc)	Attention à l'utilisation du namespace std un peu partout dans le code. La structure est bonne mais mérite peut être une petite réflexion avant de passer au rendu 2 pour harmoniser le tout (pour la hiérarchie de classe notamment). Ils y a quelques petits soucis à corriger mais vous êes sur la bonne voie.	**8**
355779	**1**	[A3] shape.h inclus message.h. Bien faire attention, shape doit rester independant de TOUT les autres modules.	**0**	[C1]particule.h,particule,particules [C1]robot.h,robot,nbUpdate Attention au attribut public, ils sont interdit. A changer pour le prochain rendu	**4**	[L1]simulation.cc56,24,25,26,robot.cc59,60,...	Attention au melange de style d`indentation et d`ouverture/fermeture de crochet	1.Pas besoin de ceer un .h s`il n`est pas necessaire. (projet.h) 2.Les variables static ne sont pas une erreur, mais attention a ne pas en abuser. Pour votre automate de lecture, dans beaucoup de cas elles pourraient être remplacée par de simple variables locales. 3.Plutôt que d`utiliser des pairs, pourquoi ne pas mettre un attribut de plus dans Robot, ou alors utiliser du polymorphisme. Beaucoup de commentaires, mais dans l`ensemble c`est vraiment du bon travail.	**5**

355797	**0**	[A3]: message.h dans shape.h [A2] robot.h dans particules.h	**3**	ok	**4**	[L1] surindentation: simulation.h 31, simulation.cc 52, shape.cc 65, robot.cc etc missing indentation: shape.cc 33-34, 37-40,51-53, 59, WARNING: [L2] robot.h	projet.h ne sert à rien	C'est globalement un bon travail, continuez comme ça	**7**
355807	**2**	ok	**0**	[C2]:particule.h,Particule(), robot.h, Robot(),RobotSpatial(): definition doit être externe, à corriger	**4**	[L1] robot.h: 15,17,29,36,58,63; particule.h: 15,17; simulation.h: 14,19	Il ne faut pas indenter les mots-clés public, private et protected	Bien mais le format des .h est à retravailler	**6**
355809	**2**	WARNING: include inutile de message.h dans robot.h et particule.h	**3**	ok	**5**	[L2] : robot.h: 64; robot.cc: 25,95	WARNING: Dépassement de 87 caractères Pas pénalisé car moins de 4 fois	Globalement, votre code est assez clair et respecte l'architecture imposée. Bon travail	**10**
355873	**2**	OK	**3**	OK	**5**		OK	Rien à redire, vous avez bien pensé à votre implémentation et votre code utilise bien les concepts vu en cours. Continuez ainsi pour le rendu 2.	**10**
355884	**2**	ok	**3**	WARNING:using namespace std ne devrait pas apparaitre les .h	**3**	[L1] particule.h: 7,11; particule.cc; robot.h: 12,14,18;simulation.h: 11,14 [L2]: robot.cc: 18,40,41,43; robot.h: 30,45,46	Il ne faut pas indenter les mots-clés public, private et protected;il ne peut y avoir 2 indentations (particule.cc), plusieurs dépassements de 87 caractères dans robot.cc et robot.h	Bien mais il y a des erreurs de mise en forme à corriger	**8**

355888	**2**	Top	**2**	[C1] robot.cc, particule.cc: in the verif methods, the vectors of the class Simulation are passed by reference and not const reference, ie they are now editable outside the class itself Suggestions: 1. shape.h: epsil_zero, dontUseEpsilZero should be in a named namespace	**5**	Top	Top Suggestions: 1. Avoid using using namespace std, utilize only what you need, eg using std::vector and so on..	The code well written and easy to understand, good job! Suggestions: 1. In shape.cc, distanceTwoPoints is contained in a namespace without apparent reasons 2. In the class Simulation, in order to avoid useless copies and large arrays, use arrays of unique_ptr of robots 3. Avoid passing around the vector containing the robots / particules (even after correcting the missing const), as it is easy to break another time the encapsulation, move the for loops in simulation and let the testing function between two robots, two particules... in the submodules 4. In project.cc, there some useless includes	**9**
355924	**2**	ok	**3**	ok	**4**	[L1] missing indentation: simulation.cc 126, shape.cc 63 double indentation: robot.cc 74, particule.cc 27-31	simulation.cc switch case: la boucle for devrait être en dehors du switch (ou dans un case)	C'est bien, continuez comme ça, vous êtes sur la bonne voie pour la suite du projet. Revoyez un peu les classes avant d'entamer la suite.	**9**
355943	**2**	WARNING: projet.cc n'a besoin que de simuation.h et shape.cc n'a pas besoin des headers de robot et particule; inutile de refaire des includes qui snt déjà dans le header du module (ex: particule, robot); le header de robot n'a pas besoin du header de particule. il n'y a pas de pénalité car aucun contenu des headers inutiles n'est utilisé. A ENLEVER.	**2**	[C2]robot.h n'externalise pas la définition des méthodes; WARNING: particule.h : faire tenir les exceptions autorisées sur une seule ligne (cf conventions)	**3**	[L1]shape.cc: 14-16, 25-27; robot.cc: 15-23,107,132,157,[L2]shape.h: 31-33, shape.cc:10,21,31,...etc ;	règles d'indentations inégalement suivies: double indentation dans shape.cc, pas d'indentation dans projet.cc, nombre d'espaces variables et différents style d'accolades dans robot.cc ; plusieurs lignes dépassent la limite acceptée de 87 caractères. Warning alignement : simulation.cc: 40	globalement la décomposition est bonne mais revoyez la présentation de l'ensemble du code car l'indentation est très inégale.	**7**

355944	**2**	Il est imposé par la donnée d'utiliser des classes pour les modules du modèle, hors Simulation n'est pas une classe. C'est à changer pour les rendu suivant. Les autre modules sont ok.	**2**	[C1]particule.cc if you want to use variable local to the module, use the keyword static warning : it is ok to have static variable in private, but try to limit them as much as possible. Don't forget to use the keyword "virtual" where appropriate to achieve proper polymorphsim.	**5**		Bien écrit.	Trop de commentaire pas toujours pertinent. Le lecteur sera aussi un proggrammeur, pas besoin d'expliquer chaque ligne de code. Un commentaire par méthode/fonction suffit généralement. Les fonctions de lecture pourrait être implémenté directement dans leur classe respective à l'aide du mot-clé static. SInon Bon travail dans l'ensemble. Bon repect des conventions notamment.	**9**
355958	**2**	Il est imposé par la donnée d'utiliser des classes pour les modules du modèle, hors Simulation n'est pas une classe. C'est à changer pour les rendu suivant. Les autre modules sont ok.	**0**	[C1]simulation.cc, robot.h, particules.h Global variable and public attribute are strictly forbidden. All attribute should only be accessed by getter/setter and not directly modified.	**5**		Pas de violation le style est bon, bravo	Pour le rendu 2 il faudra changer votre architecture pour faire disparaître les global et les attibuts publics. Cependant, le reste à un bon fond. Bon respect des conventions !	**7**
355973	**2**	OK	**3**	OK	**4**	[L2] robot.cc:144,176,180,251	Quelques oublis pour le wrapping.	Attention à l'utilisation du namespace std, sinon le project est très bon, on voit une bonne réflexion et utilisation des principes de la programmation orientée objet. C'est prometteur pour le rendu 2.	**9**
356107	**2**	ok	**3**	ok	**5**	ok		Mettre des noms de vecteurs parlant (robot et particles). Bonne mise en forme	**10**
356113	**2**	[A1] ATTENTION: faites passer l'appel du message de succès dans le modèle (simulation), il ne peut pas être contenu dans projet.cc. Ne pas importer message.h.	**3**	OK	**5**	OK	Attention aux inconsistances (std:: & std ::, nom de fonctions en majuscules dans shape, etc)	Excellent rendu! Corrigez simplement l'utilisation de message dans projet, et faites un petit effort pour la consistance du style.	**10**
356118	**1**	[A1] projet.cc only one function has to be called	**1**	[C2] robot.h, particule.h [C1] global variable: particule.cc 12, 13	**5**	ok	WARNING: shape.cc 28 etrange indentation	C'est globalement un bon travail, continuez comme ça	**7**
356263	**2**	ok	**3**	ok	**4**	[L1]simulation26-35, 37-43, etc...	concernant switch nous autorisons 2 variantes (avec case indenté ou pas) ; dans votre cas il manque l'indentation des instructions contrôlées par rapport au case. Warning alignements: shape.cc:20-21,26-27, particule.cc: 45, 56-57, 77-78, robot.cc: 58-59,91-92, etc... => à améliorer	bonne décomposition mais améliorer les alignement dans tous les modules ; inutile de ré-inclure un header qui est déjà dans le header du module (particule.cc, robot.cc).pour le rendu2, il faudra pense à ré-initialiser les variable locales static en fin de lecture.	**9**

356278	**1**	[A3] inclusion de constantes.h dans shape Warning: Eviter d'inclure des modules que vous n'utilisez pas (ex: main.cc inclusion de particule.cc et robot.cc) ou d'inclure des modules dans le header qui pourraient que figurer dans le .cc (ex: particule.h inclusion de string et constantes)	**2**	[C1] robot.h:24 la hitbox robot devrait être protected	**3**	[L2] shape.h: 26,33,53,45,etc... [P2] simulation.cc:decodage_ligne	Le module shape ne respecte pas les conventions d'indentations. La fonction decodage_ligne est trop grande (max:80) penser à la refactoriser pour le prochain rendu.	Dans l'ensemble code est bien écrit et clair. C'est vraiment dommage pour les quelques fautes d'inattention qui vous enlève quelques points. Sinon super travail.	**6**
356321	**2**	ok	**3**	ok	**4**	[L1]shape.cc	2 espace d'indentations	Attention à l'indentation. Bon code autrement	**9**
356342	**2**	OK	**3**	OK	**5**		OK	Attention à ne pas oublier de retirer le code que vous avez mis pour faire vos propres tests. Il faudra aussi sortir vos vecteurs du decodage. Attention aussi à votre utilisation de epsilon zero pour le rendu 2. A part ces petits réglages, l'approche et l'exécution sont très bonnes.	**10**
356345	**2**	OK	**3**	OK	**5**		OK	Rien à redire, vous avez bien pensé à votre implémentation et votre code utilise bien les concepts vu en cours. Continuez ainsi pour le rendu 2.	**10**
356420	**2**	Warning: Evitez d'inclure des module non-utilisé dans les header (ex: 16-17:robot.h, 9-12:shape.h, etc...), ces inclusions devraient être faites dans le .cc	**3**	OK	**5**		OK	Beau travail! Le code est bien structuré et très lisible. Deux critiques: certains noms de variable ne sont pas très clair (ex: particule L29:particule.h qui serait mieux représentée par une hitbox ou une particuleShape). Essayez aussi d'être constant dans vos noms, vous basculer de camel_case à snake_case. Sinon comme dit, bravo!	**10**
356471	**2**	Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	ok	**4**	[L1] robot.h: 15,19,25,32,43,46,53,60; particule.h:13,23	Il ne faut pas indenter les mots-clés public, private et protected	Code lisible. Globalement, l'architecture imposée est bien respectée.	**9**
356520	**2**	ok	**3**	WARNING: Simulation doit être une classe, les nombreux unamed namespace ne font que compliquer le programme	**5**	[L2]robot.cc144,163		Code dur à suivre entre les namespace et les classes avec le meme nom.	**10**
356618	**2**	Warning: inclusion inutile de shape.h dans simulation.h; inclusion inutile de cmath et message.h dans robot.h; inclusion inutile de message.h et constante.h dans particule.h; À ENLEVER ET À METTRE DANS LES .cc correspondants	**3**	ok	**4**	[L2] robot.h: 16-18; robot.cc: 5-10-16-55; simulation.cc :136-141.....	Plusieurs lignes dépassent la limite acceptée de 87 caractères (mauvais réalignement après les retours à la ligne)	Globalement, votre code est assez clair et respecte l'architecture imposée.	**9**

356756	**2**	ok	**2**	[C1] global variable: shape.cc 10, 20, particule.h 26	**4**	[L1] surindentation robot.h 63, robot.cc 42, 203 missing indentation: robot.cc 89, 124-129, 142		Oui c'est bien, mais faites attention aux variables globales	**8**
356765	**2**	WARNING: particule ne devrait pas dépendre de shape, relire la donnée	**3**	ok	**4**	[L1] particule.h: 12; robot.h: 13,19,29,34,39; simulation: 10,20	Il ne faut pas indenter les mots-clés public, private et protected	Bien	**9**
356773	**1**	[A1] Le module projet ne gère que les arguments de la ligne de commande (argc et argv). Seule une méthode de simulation doit être appelée. Le module message ne peut pas être inclus dans projet. Warning: include inutile de s headers dans spatial_robot.h	**3**	ok	**5**		ok	Très bon travail, code très lisible, bonne décomposition.	**9**
356775	**2**	ok	**3**	Bon usage des classes, l'utilisation des getter est bien réalisée. Attention cependant à l'ordre des méthodes dans le .cc et le .h, Essayez de regrouper les getters/setter au même endroit.	**2**	[L1]projet.cc28,29,35-37,shape.cc26,56-61,76-78,... [L2]robot.h43,particule.cc29,37,45,robot.cc93,107,123 [P2]simulation.cc:Remplissage et simulation	L'indentation est assez irrégulière. attention à ne pas indenter trop vos condition. Vous avez fait l'effort de faire un retour à la ligne sur vos ligne trop longue, mais dans certain cas elle dépassent quant même la limite.Le code est cependant assez agréable à lire dans l'ensemble	Bonne décomposition. Dans le module simulation, vous utilisez une méthode simulation qui devrait être écrite avec un S majuscule pour servir de vrai constructeur. Sinon bon travail !	**7**
356840	**0**	[A1] Pourquoi créer un robot spatial avant de faire la lecture ? Mettre le message de success dans lecture aussi pour le rendu 2. [A2] particule et robot ne doivent pas être inter-dépendants	**3**	Ok	**5**		ok	Le code est très bon, mais quelques erreurs dans l'idée même de l'implémentation sont a corriger pour le rendu 2.	**8**
356865	**2**	ok	**3**	Bon usage des classes, l'utilisation des getter est bien réalisée. Attention cependant à l'ordre des méthodes dans le .cc et le .h, Essayez de regrouper les getters/setter au même endroit.	**2**	[L1]projet.cc28,29,35-37,shape.cc26,56-61,76-78,... [L2]robot.h43,particule.cc29,37,45,robot.cc93,107,123 [P2]simulation.cc:Remplissage et simulation	L'indentation est assez irrégulière. attention à ne pas indenter trop vos condition. Vous avez fait l'effort de faire un retour à la ligne sur vos ligne trop longue, mais dans certain cas elle dépassent quant même la limite.Le code est cependant assez agréable à lire dans l'ensemble	Bonne décomposition. Dans le module simulation, vous utilisez une méthode simulation qui devrait être écrite avec un S majuscule pour servir de vrai constructeur. Pensez à l'utilisation de méthode static pour pouvoir placer les fonctions de test de collision directement dans les classes respectives. Sinon bon travail !	**7**
356869	**2**	ok	**3**	ok	**4**	[P2]simulation.cc92	Surindentation dans les classes: public et private ne doivent pas être indentés		**9**

356944	**2**	WARNING: projet.cc n'a besoin que de simuation.h et shape.cc n'a pas besoin des headers de robot et particule; inutile de refaire des includes qui snt déjà dans le header du module (ex: particule, robot); le header de robot n'a pas besoin du header de particule. il n'y a pas de pénalité car aucun contenu des headers inutiles n'est utilisé. A ENLEVER.	**2**	[C2]robot.h n'externalise pas la définition des méthodes; WARNING: particule.h : faire tenir les exceptions autorisées sur une seule ligne (cf conventions)	**3**	[L1]shape.cc: 14-16, 25-27; robot.cc: 15-23,107,132,157,[L2]shape.h: 31-33, shape.cc:10,21,31,...etc ;	règles d'indentations inégalement suivies: double indentation dans shape.cc, pas d'indentation dans projet.cc, nombre d'espaces variables et différents style d'accolades dans robot.cc ; plusieurs lignes dépassent la limite acceptée de 87 caractères. Warning alignement : simulation.cc: 40	globalement la décomposition est bonne mais revoyez la présentation de l'ensemble du code car l'indentation est très inégale. rajoutez aussi les header guard à vos fichiers .h.	**7**
356957	**2**	Top Warnings: 1. Do not include useless headers (e.g. project.cc)	**3**	Suggestions: 1. Even though passing by value the arrays of robots and particules in other function outside simulation doesn't strictly break encapsulation, it is very easy to do mess it up, simply move the for loop in Simulation and a create a function that check the superposition between two particules or two robots	**5**	Top	Warning: 1. Enums must be all upper case 2. Strangely all program's exit are written as std :: exit(...), you don't need the space, simply write std::exit(...)	The code is well written and easy to understand, good job! 1. Before uploading the project remove all .o files and executables 2. Instead of using shortened names, simply write the entire name, so you don't need to add comments (superpose_rn, superpose_pr...) 3. In shape1.cc, there is need for EPSIL_ZERO to be static, simply make it constrexpr 4. Store all the robots and particules as unique_ptr in order to avoid big vectors / useless copies 5. As already written try to fix the potential encapsulation problem	**10**
357156	**1**	[A3]"constants.h" inclus dans "shape.cc", à enlever	**3**	OK	**5**		WARNING - Ne pas indenter les termes "public/private"	Très bon travail globalement	**9**
357228	**2**	WARNING: un fichier unique constantes.h devrait être utilisé; le fichier principal devrait plutôt s'appeler projet; plus de tâches devraient être deleguées aux sous-modules	**3**	ok	**5**		ok	Bien mais l'architecture est à retravailler	**10**
357239	**2**	OK	**3**	OK	**4**	[L2] robot.cc:144,176,180,251	Quelques oublis pour le wrapping.	Attention à l'utilisation du namespace std, sinon le project est très bon, on voit une bonne réflexion et utilisation des principes de la programmation orientée objet. C'est prometteur pour le rendu 2.	**9**
357419	**2**	ok	**3**	ok	**5**		ok	Magnifique	**10**
357437	**2**	ok	**0**	[C2]particule robot, simulation	**5**	ok	Warning: attention à la consistance des espaces autour de ::	Tout mettre à l'intérieur des header guards. Une methode peut être définie uniquement si elle est écrite sur la même ligne que la déclaration	**7**

357574	**1**	[A3] shape n'a pas connaissance des robots ni particules, que des formes primitives	**3**	OK	**5**	[L2] robot.cc:43,76,169;robot.h:85	Petits oublis dans le wrapping.	Très bon code, bien structuré et utilisant intelligemment les principes de la programmation orientée objet. Il faut juste corriger la nomenclature dans votre module shape. Continuez ainsi pour le rendu 2.	**9**
357595	**2**	OK	**3**	OK	**5**		OK	Rien à redire, vous avez bien pensé à votre implémentation et votre code utilise bien les concepts vu en cours. Continuez ainsi pour le rendu 2.	**10**
357650	**1**	[A1] Le module projet n'a que le droit d'appeler le sous-module simulation	**2**	[C2] 10:particule.h Le constructeur doit être externalisé	**3**	[L1] robot.h, shape.h, particule.h [L2] robot.cc: 14-15,17-18,24-25,28-29,36-37	Attention à ne pas indenter public et private dans vos header. Plusieurs lignes dépasse la limite de caractère dans robot.cc	L'idée de créer une liste de hitbox pour généraliser les collisions est intéressante mais malheureusement mal exécuté. Evitez de donner trop de fonctionnalités à une classe, je pense notamment à shape qui s'occupe à la fois d'être un carré, un cercle et une zone. Vos fonctions de lecture dans simulation et de superpositions dans shape mériterait d'être refactorisez car elle sont dûr à lire. Dernièrement pour les rendus suivants je vous conseille de changer votre manière de stocker les entités de la simulation (pensez à créer une classe simulation), car dans une liste de hitbox on perd l'information sur quelle entités sont en collisions.	**6**
357744	**2**	Ok	**3**	Ok	**5**		Ok	Excellent rendu, alignez bien vos arguments lors de retours à la ligne sur les listes d'arguments	**10**
357752	**2**	Bonne architecture.	**3**	ok	**5**		Très bien écrit, bravo. Quelques rares variablse auraient cependant pu avoir des noms un peu plus parlant	Code très agréable à lire, identation parfaite, commentaire et ligne de séparation aidant bien à la compréhension, très bon travail !	**10**

357754	**2**	Il est imposé par la donnée d'utiliser des classes pour les modules du modèle, hors Simulation n'est pas une classe. C'est à changer pour les rendu suivant. Les autre modules sont ok.	**2**	[C1]particule.cc if you want to use variable local to the module, use the keyword static warning : it is ok to have static variable in private, but try to limit them as much as possible. Don't forget to use the keyword "virtual" where appropriate to achieve proper polymorphsim.	**5**		Bien écrit.	Trop de commentaire pas toujours pertinent. Le lecteur sera aussi un proggrammeur, pas besoin d'expliquer chaque ligne de code. Un commentaire par méthode/fonction suffit généralement. Les fonctions de lecture pourrait être implémenté directement dans leur classe respective à l'aide du mot-clé static. SInon Bon travail dans l'ensemble. Bon repect des conventions notamment.	**9**
357841	**1**	[A3] shape ne doit pas dépendre de constantes, c'est un module indépendant du reste de votre projet	**3**	Ok	**4**	[L1]robot.cc:54	Plusieurs erreurs d'indentation dans vos blocs imbriqués qui rendent la lecture de votre code moins agréable	Très bon rendu, faites attention à votre indentation	**8**
357849	**2**	ok	**3**	ok	**5**	Warning: [L1]: missing indentation simulation.cc 172,		C'est bien, continuez comme ça, vous êtes sur la bonne voie pour la suite du projet. Revoyez un peu les classes avant d'entamer la suite.	**10**
357857	**2**	ATTENTION: le cout du message de succès devrait se trouver dans le modèle et non dans le module projet, et devrait utiliser le module message et non un string écrit par vous-mêmes.	**3**	OK	**5**	OK	Soyez consistants pour vos noms de fichiers (ne pas mélanger minuscules/majuscules). Un tel mélange peut mener à des erreurs de compilation si on ne fait pas attention...	Bon respect de l'architecture en dehors du point mentionné, utilisation des classes et styles impeccables. Excellent travail!	**10**
358190	**2**	ATTENTION: le cout du message de succès devrait se trouver dans le modèle et non dans le module projet, et devrait utiliser le module message et non un string écrit par vous-mêmes.	**3**	OK	**5**	OK	OK	Bon respect de l'architecture en dehors du point mentionné, utilisation des classes et styles impeccables. Excellent travail!	**10**
358257	**2**	OK	**3**	OK	**4**	[L1]shape.cc, particule.cc31	Mauvaises indentations	Très bon travail	**9**
358300	**2**	ok	**3**	ok	**4**	[L2] simulation.h 33, simulation.cc, robot.h etc		C'est globalement un bon travail, continuez comme ça	**9**
358305	**1**	[A1] projet.cc only one function has to be called	**1**	[C2] robot.h, particule.h [C1] global variable: particule.cc 12, 13	**5**	ok	WARNING: shape.cc 28 etrange indentation	C'est globalement un bon travail, continuez comme ça	**7**

358312	**2**	ok	**3**	ok	**4**	[L1] simulation.h24-27, particule.h18-21,...	Indentation peu consistante,surindentation des les calsses: public et private ne doivent pas être indentés	Mettre les includes entre les header guards, attention à l'indentation. Bon code autrement	**9**
358313	**2**	Ok	**3**	Ok	**4**	[L1] particule.cc:34	Plusieurs erreurs d'indentation qui rendent la lecture de votre code moins agréable	Très bon rendu, faites attention à votre indentation	**9**
358420	**2**	OK	**3**	OK	**5**	[L1] warning particule.cc: 15,20 [L2] warning robot.h:63 robot.cc:94,119	Attention aux erreurs d'indentation et à la limite de caractère. Quand vos lignes sont trop longues essayer de les aligner par rapport au début des paramètre de la ligne du dessus.	Le code est bien pensé et très lisible. Bravo!	**10**
358442	**2**	Ok	**3**	Ok	**5**		Ok	Excellent rendu, rien à redire!	**10**
358456	**2**	ok	**3**	ok	**5**	ok		Pourquoi passer l'instance de simulation a simulation ? Bonne décomposition autrement	**10**
358476	**2**	ok	**2**	[C1] global variable: shape.cc 10, 20, particule.h 26	**4**	[L1] surindentation robot.h 63, robot.cc 42, 203 missing indentation: robot.cc 89, 124-129, 142		Oui c'est bien, mais faites attention aux variables globales	**8**
358500	**2**	ok	**0**	[C2]particule robot, simulation	**5**	ok	Warning: attention à la consistance des espaces autour de ::	Tout mettre à l'intérieur des header guards. Une methode peut être définie uniquement si elle est écrite sur la même ligne que la déclaration	**7**

359179	**2**	Top Warnings: 1. Do not include useless headers (e.g. project.cc)	**3**	Suggestions: 1. Even though passing by value the arrays of robots and particules in other function outside simulation doesn't strictly break encapsulation, it is very easy to do mess it up, simply move the for loop in Simulation and a create a function that check the superposition between two particules or two robots	**5**	Top	Warning: 1. Enums must be all upper case 2. Strangely all program's exit are written as std :: exit(...), you don't need the space, simply write std::exit(...)	The code is well written and easy to understand, good job! 1. Before uploading the project remove all .o files and executables 2. Instead of using shortened names, simply write the entire name, so you don't need to add comments (superpose_rn, superpose_pr...) 3. In shape1.cc, there is need for EPSIL_ZERO to be static, simply make it constrexpr 4. Store all the robots and particules as unique_ptr in order to avoid big vectors / useless copies 5. As already written try to fix the potential encapsulation problem	**10**
359187	**2**	Warning: inclusion inutile de message.h dans robot.h et dans particule.h; inclusion inutile de shape.h dans simulation.h	**3**	ok	**5**		WARNING alignments: robot.h: 28-32,36-41,97; robot.cc:34,45,46....	Très bonne décomposition. Néanmoins, il y a quelques imprécisions dans la présentation de votre code (réalignez vos instructions après les retours à la ligne)	**10**
359285	**2**	OK	**2**	[C1] particule.h:19 list_particule	**4**	[L1] simulation.cc: 36-39, 61-64, 68,81	Attention aux doubles indentations dans les switch	Super code! C'est juste un peu dommage pour les erreurs bêtes qui vous font perdre quelques points. Bon travail!	**8**
359444	**2**	Ok	**3**	Ok	**4**	[L1] simulation.cc:78	Quelques erreurs d'indentation sur les instructions composées et dans les déclarations de vos classes qui rendent la lecture de votre code moins agréable	Excellent rendu! Quelques légères erreurs sur les indentations à corriger qui pourraient rendre votre code encore plus lisible	**9**
360558	**1**	No class for simulation, and projet.cc include robot.h.	**3**	Bon usage des static interne au module, attention cependant à ne pas en abuser	**5**		Une ou deux petites erreurs d'identation, mais sinon très bien	Très bon code dans l'ensemble, essayez de corriger les deux petite erreus d'architecture pour le rendu 2 et ça sera très bien.	**9**
360837	**1**	[A3] shape ne doit pas dépendre de constantes, c'est un module indépendant du reste de votre projet	**3**		**4**	[L1] robots.cc:70	Quelques erreurs d'indentation et d'inconsistence qui rendent la lecture de votre code moins agréable	Très bon rendu, quelques erreurs d'indentation	**8**

360931	**2**	OK	**1**	[C2]Définition des méthodes à externaliser; ne pas définir de méthodes dans l'interface des modules. C'est le cas pour robot.h, particule.h	**5**		WARNING - Ne pas indenter les termes "public/private"	Très bon travail globalement, réparer les 2 interfaces de module spécifiées pour le prochain rendu	**8**
360970	**2**	Bonne architecture. Faites attention à n'inclure dans les .cc uniquement ce qui n'est pas déjà inclu dans le .h. Essayez aussi d'inclure au maximum les choses uniquement dans le .cc si cela n'est pas nécessaire dans le .h.	**3**	Bon usages des classes ainsi que des getter/setter.	**4**	[L1]robot.cc25,27,robot.h24,30,37,46,... simulation.cc23,25	Beau code dans l'ensemble, attention cependant, on ne doit pas identer les public/private dans les fichier.h	Bon travail le code est propre et bien réalisé. Attention cependant, dans le module simulation, la methode initialisation pourrait être incluse directement dans le constructeur.	**9**
361004	**2**	WARNING: particule et robot ne devraient pas dépendre de shape, relire la donnée	**3**	WARNING:using namespace std ne devrait pas apparaitre les .h	**4**	[L2] :robot.cc: 19,29,39,48,58	Plusiseurs dépassement de 87 caractères dans robot.cc et shape.cc, WARNING: indentation devrait être plus régulière, WARNING: lecture devrait être divisée	Globalement correct	**9**
361007	**2**	OK	**3**	OK	**3**	[L1] robot.cc: 92,99,94,138,147,etc... [L2] warning: plusieurs ligne dans robot.h sont à 88 caractère [P2] simulation.cc:decodage_ligne	Attention les règles l'indentations ne sont pas respectées dans robot.cc. Faites attention aux wrapping vous avez plusieurs lignes à 88 caractères. La fonction de décodage est malheuresement trop longue, pensez à la refactoriser pour le prochain rendu	Le code est bien écrit et clair (à part un petit dérapage dans robot.cc). Vos fonctions addRobot/addParticule portent plusieurs chapeaux, elles s'occupent à la fois d'ajouter l'entité aux vector respectifs, de tester d'éventuelles collision et de print un message d'erreur. Essayez de garder un but unique pour vos fonctions. Bon boulot!	**8**
361030	**2**	OK	**3**	OK	**5**		OK	Attention à ne pas oublier de retirer le code que vous avez mis pour faire vos propres tests. Il faudra aussi sortir vos vecteurs du decodage. Attention aussi à votre utilisation de epsilon zero pour le rendu 2. A part ces petits réglages, l'approche et l'exécution sont très bonnes.	**10**
361046	**2**	OK	**3**	OK	**5**		OK	Super code! Très lisible et bien pensé! Dans le module robot vos fonctions de superpositions vous passez par valeur le vecteur de particule. Pour de grande simulation cela pourrait vous coûter en performance.	**10**
361051	**2**	Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	ok	**4**	[L1] robot.h: 15,19,25,32,43,46,53,60; particule.h:13,23	Il ne faut pas indenter les mots-clés public, private et protected	Code lisible. Globalement, l'architecture imposée est bien respectée.	**9**

361066	**0**	[A1]le module projet à remplacé le module simulation ce qui n`est pas conforme du tout à la donné. [A3]shape est dépendant de constante.h, ce qui n`est pas conforme. Warning1: Bien relire la donné et corriger l`architecture du programme avant le prochain rendu. Warning2: Posez la question a un TA ou Mr.Boulic lors de la prochaine séance d`exercice pour demander l`authorisation d`avoir des modules supplémentaires non specifié par la donnée.	**3**	Ok	**3**	[L2]projet.cc53, robot.h104,112, robot.cc22,57,... [P2]robot.cc, ajouterparticule, ajouterRobotNeutraliseur	Attention aux grosses fautes d'orthographe dans les noms de variables, ce n'est pas très propre(veteur,deistance,...). Pensez a vérifier avant le rendu si votre code ne dépasse pas la limite des 87 charactères sur geany avant le rendu.	Travail Ok dans l'ensemble. L'architecture est cependant vraiment à retravailler pour le prochaion rendu. N'hésitez pas à demander de l'aide à un.e assistant.e si nécessaire.	**6**
361116	**1**	[A3] Faites attention à ne pas inclure de module de plus haut niveau dans shape (6-9:shape.cc) Warning: Eviter d'inclure des modules non-utilisé dans les header, ces inclusions devraient être faites dans le .cc (ex:4,5 dans particule.h) Warning: Vous incluez souvent des modules que vous n'utilisez pas (ex:robot.g ligne 4,5)	**3**	OK	**4**	[L2] 22:particule.cc; 33:simulation.cc; 29, 33:shape.h; 13,15,29:shape.cc,etc...	Vous avez oubliez la limite de ligne à 87 caractère	Globalement le code est bon. Vous avez fait un effort pour le rendre bien lisible. C'est dommage que vous ayez perdu des points un peu bêtement avec la limite de ligne et les include inutiles. Bon travail!	**8**
361124	**0**	[A1]le module projet à remplacé le module simulation ce qui n`est pas conforme du tout à la donné. [A3]shape est dépendant de constante.h, ce qui n`est pas conforme. Warning1: Bien relire la donné et corriger l`architecture du programme avant le prochain rendu. Warning2: Posez la question a un TA ou Mr.Boulic lors de la prochaine séance d`exercice pour demander l`authorisation d`avoir des modules supplémentaires non specifié par la donnée.	**3**	Ok	**3**	[L2]projet.cc53, robot.h104,112, robot.cc22,57,... [P2]robot.cc, ajouterparticule, ajouterRobotNeutraliseur	Attention aux grosses fautes d'orthographe dans les noms de variables, ce n'est pas très propre(veteur,deistance,...). Pensez a vérifier avant le rendu si votre code ne dépasse pas la limite des 87 charactères sur geany avant le rendu.	Travail Ok dans l'ensemble. L'architecture est cependant vraiment à retravailler pour le prochaion rendu. N'hésitez pas à demander de l'aide à un.e assistant.e si nécessaire.	**6**
361133	**2**	ok	**3**	ok	**4**	[L1] simulation.h24-27, particule.h18-21,...	Indentation peu consistante,surindentation des les calsses: public et private ne doivent pas être indentés	Mettre les includes entre les header guards, attention à l'indentation. Bon code autrement	**9**

361150	**2**	ok	**3**	ok	**4**	[L1]simulation26-35, 37-43, etc...	concernant switch nous autorisons 2 variantes (avec case indenté ou pas) ; dans votre cas il manque l'indentation des instructions contrôlées par rapport au case. Warning alignements: shape.cc:20-21,26-27, particule.cc: 45, 56-57, 77-78, robot.cc: 58-59,91-92, etc... => à améliorer	bonne décomposition mais améliorer les alignement dans tous les modules ; inutile de ré-inclure un header qui est déjà dans le header du module (particule.cc, robot.cc).pour le rendu2, il faudra pense à ré-initialiser les variable locales static en fin de lecture.	**9**
361164	**2**	ok, module suppémentaire pertinent.	**0**	[C1]particule et robot. La modification des attributs de simulation (les liste de particules et robot) se fait par reference avec des fonctions d`autre modules(...init), ce qui viole le principe d`encapsulation.	**5**		Beau code, agréable à lire et respectant bien les conventions	Bravo pour l`organisation de votre code et l`excellent respect des conventions. Cependant, il faudra modifier pour le prochain rendu la manière de modifier les attributs de simulation, en utilisant par exemple des setters	**7**
361174	**0**	[A1] projet ne doit dépendre que de simulation, [A3] shape ne doit pas dépendre de constantes	**3**	Ok	**5**		Ok	Bon rendu, faites attention à respecter la hiérarchie des modules	**8**
361235	**2**	ok	**3**	ok	**5**	ok		parfait	**10**

361276	**2**	Top Warnings: 1. Remove double includes in both the header and .cc files	**3**	Top	**5**	Top	Top Suggestions: 1. Instead of using using namespace std, utilize only what you need, eg: using std::vectors 2. Stick to one variable naming style: snake_case vs camelCase 3. Utilize pointers and reference to pass struct and classes in order to avoid useless copy 4. Some methods could have been named better: sup_rop_part... so you don't need to add a comment to explain their functionality	The code is well written and easy to understand, good job! Suggestions: 1. In the class Simulation, use vector of unique_ptr of Robots and Particules, in order to avoid useless copies 2. Divide the Robot class in smaller classes 3. In the class Simulation, the method lecture is a little bit verbose, you could divide it in two functions on the reads the data and one after that controlls all the superpositions 4. Decide between camelCasing and snake_casing	**10**
361316	**2**	Ok	**3**	Ok	**5**		Ok	Excellent rendu, quelques remarques: les noms des fonctions ne sont pas très descriptifs et vos aérations ne sont pas consistentes	**10**
361362	**2**	Bonne architecture. Faites attention à n'inclure dans les .cc uniquement ce qui n'est pas déjà inclu dans le .h. Essayez aussi d'inclure au maximum les choses uniquement dans le .cc si cela n'est pas nécessaire dans le .h.	**3**	Bon usages des classes ainsi que des getter/setter.	**4**	[L1]robot.cc25,27,robot.h24,30,37,46,... simulation.cc23,25	Beau code dans l'ensemble, attention cependant, on ne doit pas identer les public/private dans les fichier.h	Bon travail le code est propre et bien réalisé. Attention cependant, dans le module simulation, la methode initialisation pourrait être incluse directement dans le constructeur.	**9**

361380	**2**	Top	**3**	Suggestions: 1. In particule.cc, robot.cc, the vectors of robots and particules are passed as const reference, even if it doesn't break the encapsulation, it is not good practice as if you onetime forgot const, then it breaks the encapsulation	**4**	[L1] robot.cc: the indent space is uneven sometimes 3 spaces, sometimes 4	Warnings: 1. Inconsistent naming style for variables/methods: camelCase or snake_case 2. One stament per line 3. Inconsistent spacing: use a formatting tool 4. Sometimes the names of variables/methods are not clear enough 5. Switch cases need brackets only if temporary variables are used (in robot.cc)	The code is readeable and understandable, there is some cleaning up left to do, but overall good job! Try to use better names and divide your function in smaller one with clearly defined goals: 1. Use vectors of pointers (or better yet unique_ptr) for classes and structs in order to avoid copies (applies also for lecture_robot... return the pointer and not the instance) 2. The test functions should return true / false especially since for the next "rendu", the program will not exit when it encounters an error in the configuration file 3. Better to avoid passing const reference of attributes around, move the for loops in Simulation and let the testing methods between two particule, two robots.. in the submodules 4. In particule.cc, lecture_particules don't pass the ifstream instance, but simply the line so you can't break the other functions by for example skipping inadvertly a line 5. Divide the class Robot in multiple subclasses	**9**

361405	**2**	Ok Warnings: 1. Very complicated definition of the instance of Simulation, simply define it in project.cc	**2**	[C1]: simulation.cc: the class Simulation has as static attribute itself, therefore by using the method getistance that return the static attribute by reference, it is possible to modify it outside the class --> breaks the encapsulation and too complicated... simply add it as variable in project.cc Suggestions: 1. shape.h: epsil_zero should be in a named namespace	**4**	[L1] robot.h: do not indent private, public [L1] shape.cc: mixed indent spaces [L1] simulation.cc: mixed indentation style for decodage_ligne	Warnings: 1. Incosistent naming case for variables/methods: camelCase or snake_case 2. All classes must start with an uppercase 3. Overall incosistent spacing: use a code formatter! 4. Do not use typedef to define a type that you will use one time, simply use a good name for the variable 5. Try to give better names to methods/variables...	The code is not very readable and sometimes convoluted. Try to give better names to variables and methods, divide big functions into smaller functions with each a specific task: 1. In project.cc you don't need to check whether or not the first argument is equal to "./projet", as it is the name of the executable 2. In the module shape, the function distance is set as static without any apparent reason 3. As already written, fix the Simulation::gestinstance 4. Simulation::decodage_ligne is a little bit convoluted as there are: multiple different variables with similar names, functions called test but don't return nothing and instead directly modify the vectors, incosistent spacing...	**8**
361537	**1**	No class for simulation, and projet.cc include robot.h.	**3**	Bon usage des static interne au module, attention cependant à ne pas en abuser	**5**		Une ou deux petites erreurs d'identation, mais sinon très bien	Très bon code dans l'ensemble, essayez de corriger les deux petite erreurs d'architecture pour le rendu 2 et ça sera très bien.	**9**
361567	**2**	ok	**3**		**4**	[P2]simulation.cc41	attention aux espaces autour de ::	Vous avez pris un bon départ, continuez comme ça.	**9**
361757	**2**	Top Warnings: 1. Remove double includes in both the header and .cc files	**3**	Top	**5**	Top	Top Suggestions: 1. Instead of using using namespace std, utilize only what you need, eg: using std::vectors 2. Stick to one variable naming style: snake_case vs camelCase 3. Utilize pointers and reference to pass struct and classes in order to avoid useless copy 4. Some methods could have been named better: sup_rop_part... so you don't need to add a comment to explain their functionality	The code is well written and easy to understand, good job! Suggestions: 1. In the class Simulation, use vector of unique_ptr of Robots and Particules, in order to avoid useless copies 2. Divide the Robot class in smaller classes 3. In the class Simulation, the method lecture is a little bit verbose, you could divide it in two functions on the reads the data and one after that controlls all the superpositions 4. Decide between camelCasing and snake_casing	**10**

362005	**2**	ok	**3**	Les constructeurs devraient être utilisés plutôt que les setters.	**5**		ok	Joli.	**10**
362019	**2**	ok	**3**	ok	**5**		ok	Magnifique	**10**
362057	**2**	WARNING: un fichier unique constantes.h devrait être utilisé; le fichier principal devrait plutôt s'appeler projet; plus de tâches devraient être deleguées aux sous-modules	**3**	ok	**5**		ok	Bien mais l'architecture est à retravailler	**10**
362176	**1**	[A3] Faites attention à ne pas inclure de module de plus haut niveau dans shape (6-9:shape.cc) Warning: Eviter d'inclure des modules non-utilisé dans les header, ces inclusions devraient être faites dans le .cc (ex:4,5 dans particule.h) Warning: Vous incluez souvent des modules que vous n'utilisez pas (ex:robot.g ligne 4,5)	**3**	OK	**4**	[L2] 22:particule.cc; 33:simulation.cc; 29, 33:shape.h; 13,15,29:shape.cc,etc...	Vous avez oubliez la limite de ligne à 87 caractère	Globalement le code est bon. Vous avez fait un effort pour le rendre bien lisible. C'est dommage que vous ayez perdu des points un peu bêtement avec la limite de ligne et les include inutiles. Bon travail!	**8**
362199	**2**	Top	**3**	Top	**5**	Top	Top Suggestions: 1. Incosistent spacing, use a code formatter! 2. In order to avoid copy struct and vector use const reference and vectors of pointers	The code is readable and overall easy to understand, goob job! Suggestions: 1. Try centralize the ownership of the lists, for example by adding them to the class Simulation. 2. Use pointers and reference in order to avoid copying everytime structs, classes and vectors. 3. Divide the class Robot in multiple classes in order to simplify it 4. Remove .o from the project before uploading it	**10**
362201	**2**	ok	**3**	ok	**4**	[L1] missing indentation: simulation.cc 126, shape.cc 63 double indentation: robot.cc 74, particule.cc 27-31	simulation.cc switch case: la boucle for devrait être en dehors du switch (ou dans un case)	Vous avez pris un bon départ, continuez comme ça.	**9**
362204	**2**	OK	**2**	[C1] particule.h:19 list_particule	**4**	[L1] simulation.cc: 36-39, 61-64, 68,81	Attention aux doubles indentations dans les switch	Super code! C'est juste un peu dommage pour les erreurs bêtes qui vous font perdre quelques points. Bon travail!	**8**
362233	**2**	ok	**0**	[C2]:particule.h,Particule(), robot.h, Robot(),RobotSpatial(): definition doit être externe, à corriger	**4**	[L1] robot.h: 15,17,29,36,58,63; particule.h: 15,17; simulation.h: 14,19	Il ne faut pas indenter les mots-clés public, private et protected	Bien mais le format des .h est à retravailler	**6**

362253	**2**	Architecture ok	**3**	Ok, no global variable, good use of "const" in the getter	**4**	[L1]simulation.cc 47,59,62,particle.cc 15	Bon style, dommage pour les quelques petites erreurs d'identation	Bon travail beau code. Attention cependant, les switch case ne nécessitent pas forcément de bracket	**9**
362308	**1**	[A1] Le module projet ne gère que les arguments de la ligne de commande (argc et argv). Seule une méthode de simulation doit être appelée. Le module message ne peut pas être inclus dans projet. Warning: include inutile de s headers dans spatial_robot.h	**3**	ok	**5**		ok	Très bon travail, code très lisible, bonne décomposition.	**9**
362320	**2**	Il est imposé par la donnée d'utiliser des classes pour les modules du modèle, hors Simulation n'est pas une classe. C'est à changer pour les rendu suivant. Les autre modules sont ok.	**0**	[C1]simulation.cc, robot.h, particules.h Global variable and public attribute are strictly forbidden. All attribute should only be accessed by getter/setter and not directly modified.	**5**		Pas de violation le style est bon, bravo	Pour le rendu 2 il faudra changer votre architecture pour faire disparaître les global et les attibuts publics. Cependant, le reste à un bon fond. Bon respect des conventions !	**7**
362358	**2**	OK	**3**	OK	**3**	[L1] robot.cc: 92,99,94,138,147,etc... [L2] warning: plusieurs ligne dans robot.h sont à 88 caractère [P2] simulation.cc:decodage_ligne	Attention les règles l'indentations ne sont pas respectées dans robot.cc. Faites attention aux wrapping vous avez plusieurs lignes à 88 caractères. La fonction de décodage est malheuresement trop longue, pensez à la refactoriser pour le prochain rendu	Le code est bien écrit et clair (à part un petit dérapage dans robot.cc). Vos fonctions addRobot/addParticule portent plusieurs chapeaux, elles s'occupent à la fois d'ajouter l'entité aux vector respectifs, de tester d'éventuelles collision et de print un message d'erreur. Essayez de garder un but unique pour vos fonctions. Bon boulot!	**8**
362379	**2**	Ok	**3**	Ok	**4**	[L1] simulation.cc:78	Quelques erreurs d'indentation sur les instructions composées et dans les déclarations de vos classes qui rendent la lecture de votre code moins agréable	Excellent rendu! Quelques légères erreurs sur les indentations à corriger qui pourraient rendre votre code encore plus lisible	**9**
362425	**0**	[A3]: message.h dans shape.h [A2] robot.h dans particules.h	**3**	ok	**4**	[L1] surindentation: simulation.h 31, simulation.cc 52, shape.cc 65, robot.cc etc missing indentation: shape.cc 33-34, 37-40,51-53, 59, WARNING: [L2] robot.h	projet.h ne sert à rien	C'est globalement un bon travail, continuez comme ça	**7**

362429	**2**	OK	**3**	OK	**4**	[L2] particule.h:31;robot.cc:70,81;simulation.cc:142,175,183	Petits oublis dans le wrapping.	Attention à bien penser à retirer votre code de test. A part cela le projet est parfaitement réfléchi et exécuté, continuez ainsi pour le rendu 2.	**9**
362434	**2**		**3**		**5**				**10**
362471	**1**	[A3] shape ne doit pas dépendre de constantes, c'est un module indépendant du reste de votre projet	**3**	Ok	**4**	[L1]robot.cc:54	Plusieurs erreurs d'indentation dans vos blocs imbriqués qui rendent la lecture de votre code moins agréable	Très bon rendu, faites attention à votre indentation	**8**
362474	**0**	[A1] projet ne doit dépendre que de simulation, [A3] shape ne doit pas dépendre de constantes	**3**	Ok	**5**		Ok	Bon rendu, faites attention à respecter la hiérarchie des modules	**8**
362542	**2**	OK	**3**	OK	**5**		OK	Le code est très clair, bien aéré. Petite remarque: votre classe simulation est assez grande, vous pourriez peut-être gérer certains type de collisions dans les classes particule et robot et/ou utiliser le polymorphisme pour éviter de créer des fonctions répétitives.	**10**
362549	**2**	OK	**3**	WARNING:using namespace std ne devrait pas apparaitre dans les .h	**4**	[L1] simulation.h:13,15; particule.cc:59; partout [021] robot.h	WARNING: format des header guards: MON_SOURCE_H pour mon_source.h Ne pas indenter private & public (simulation.h), inconsistance accolade (particule.cc) Il semble que dans certains fichiers, vos premiers niveaux d'indentation ont un espacement de 3 caractères, tandis que les suivant ont un espacement de 4 caractères. Cela est probablement dû au mélanges espaces/TAB. À corriger, il faut que ça soit consistant.	Excellent rendu, corrigez seulement l'indentation inconsistance	**9**
362582	**2**	Ok	**3**	Ok	**5**		Ok	Excellent rendu, alignez bien vos arguments lors de retours à la ligne sur les listes d'arguments	**10**

362633	**2**	WARNING: particule et robot ne devraient pas dépendre de shape, relire la donnée	**3**	WARNING:using namespace std ne devrait pas apparaitre les .h	**4**	[L2] :robot.cc: 19,29,39,48,58	Plusiseurs dépassement de 87 caractères dans robot.cc et shape.cc, WARNING: indentation devrait être plus régulière, WARNING: lecture devrait être divisée	Globalement correct	**9**
362672	**2**	ok	**3**	ok	**5**	ok		parfait	**10**
362713	**2**	ok	**3**		**4**	[P2]simulation.cc41	attention aux espaces autour de ::	Vous avez pris un bon départ, continuez comme ça.	**9**
362819	**2**	Top	**3**	Top	**5**	Top	Top	The code is excellently written and easy to understand Suggestions: 1. In order to avoid useless copies, use vectors of unique_ptr for the robots	**10**
362863	**2**	ok	**3**	Les constructeurs devraient être utilisés plutôt que les setters.	**5**		ok	Joli.	**10**
362883	**1**	[A1] Le module projet n'a que le droit d'appeler le sous-module simulation	**2**	[C2] 10:particule.h Le constructeur doit être externalisé	**3**	[L1] robot.h, shape.h, particule.h [L2] robot.cc: 14-15,17-18,24-25,28-29,36-37	Attention à ne pas indenter public et private dans vos header. Plusieurs lignes dépasse la limite de caractère dans robot.cc	L'idée de créer une liste de hitbox pour généraliser les collisions est intéressante mais malheureusement mal exécuté. Evitez de donner trop de fonctionnalités à une classe, je pense notamment à shape qui s'occupe à la fois d'être un carré, un cercle et une zone. Vos fonctions de lecture dans simulation et de superpositions dans shape mériterait d'être refactorisez car elle sont dûr à lire. Dernièrement pour les rendus suivants je vous conseille de changer votre manière de stocker les entités de la simulation (pensez à créer une classe simulation), car dans une liste de hitbox on perd l'information sur quelle entités sont en collisions.	**6**
362932	**2**	Bonne architecture.	**3**	ok	**5**		Très bien écrit, bravo. Quelques rares variables auraientt cependant pu avoir des noms un peu plus parlant	Code très agréable à lire, identation parfaite, commentaire et ligne de séparation aidant bien à la compréhension, très bon travail !	**10**
362936	**2**	ok	**3**	ok	**5**	ok		Mettre des noms de vecteurs parlant (robot et particles). Bonne mise en forme	**10**
362937	**2**	OK	**3**	OK	**5**		ok	code impeccable ; très bon usage du principe d'abtraction	**10**

363073	**2**	[A2] WARNING: demander dès maintenant à Ronan Boulic l'autorisation d'utiliser votre architecture non-conventionnelle	**3**	ok	**5**	ok	ok	Très bon rendu, mais votre architecture doit être validée par l'enseignant avant de pouvoir continuer avec celle-ci.	**10**
363089	**2**	Warning: Evitez d'inclure des module non-utilisé dans les header (ex: 16-17:robot.h, 9-12:shape.h, etc...), ces inclusions devraient être faites dans le .cc	**3**	OK	**5**		OK	Beau travail! Le code est bien structuré et très lisible. Deux critiques: certains noms de variable ne sont pas très clair (ex: particule L29:particule.h qui serait mieux représentée par une hitbox ou une particuleShape). Essayez aussi d'être constant dans vos noms, vous basculer de camel_case à snake_case. Sinon comme dit, bravo!	**10**
363093	**2**	ok	**3**	ok	**4**	[L1] missing indentation in namespace shape.cc,, shape.h	Il y a souvent une surindentation (module robot, particule.cc), WARNING: vous utilisez une fonction Lecture dsans projet.cc qui n est pas defini et vous n'utilisez pas la fonction lecture de votre module simulation	Vous avez pris un bon départ, continuez comme ça.	**9**
363258	**2**	Warning: Include inutile de particule.h et de shape.h dans simulation.h; Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	ok	**5**	[L1] simulation.cc 20-25,26.	WARNING double indentation: Pas pénalisé car moins de 4 fois	Excellente décomposition. Code très agréable à lire mais respectez l'organisation des fichiers (ex: robot.h les typedef sont avant la définition des classes)	**10**
363301	**2**	OK	**3**	OK	**5**		OK	Super code! Très lisible et bien pensé! Dans le module robot vos fonctions de superpositions vous passez par valeur le vecteur de particule. Pour de grande simulation cela pourrait vous coûter en performance.	**10**
363358	**2**	Ok	**3**	Ok	**5**		Ok	Excellent rendu, rien à redire!	**10**
363444	**1**	[A3] shape ne doit pas dépendre de message	**3**	Ok	**5**		Ok	Excellent rendu, très bon code, le module shape ne doit pas gérer l'affichage des messages d'erreurs mais seulement la détection des erreurs	**9**
363599	**2**	WARNING: inclusion inutile de message.h dans robot.h et dans particule.h; inclusion inutile de shape.h dans simulation.h	**3**	ok	**5**		WARNING alignments: robot.h: 28-32,36-41,97; robot.cc: 34,45,46....	Très bonne décomposition. Néanmoins, il y a quelques imprécisions dans la présentation de votre code (réalignez vos instructions après les retours à la ligne)	**10**

363600	**2**	OK	**3**	OK	**5**		ok	code impeccable ; très bon usage du principe d'abtraction	**10**
363687	**2**	Ok	**3**	Ok	**5**		Ok	Excellent rendu, n'hésitez pas à séparer vos conditions multilignes en plusieurs booléens reliés ensuite par des conjonctions logiques	**10**
363718	**2**	OK	**3**	OK	**5**		OK	Le code est très clair, bien aéré. Petite remarque: votre classe simulation est assez grande, vous pourriez peut-être gérer certains type de collisions dans les classes particule et robot et/ou utiliser le polymorphisme pour éviter de créer des fonctions répétitives.	**10**
363765	**1**	[A3] shape doit être INDEPENDANT de son utilisation dans votre projet	**3**	Ok	**3**	[L1] Tout code [L2] Tout code	Vous n'avez pas fait attention au wrapping. Il ne doit pas y avoir d'indentation au niveau des mots clés public et private. Des erreurs d'indentation dans simulation.cc aussi	Votre code est assez efficace donc on voit qu'il y a eu de la réflexion mais vous avez du faire quelques erreurs à cette étape. Attention à bien respecter les consignes, mais cela peut s'arranger assez facilement pour le rendu 2.	**7**
363799	**2**	Top	**2**	[C1] robot.cc, particule.cc: in the verif methods, the vectors of the class Simulation are passed by reference and not const reference, ie they are now editable outside the class itself Suggestions: 1. shape.h: epsil_zero, dontUseEpsilZero should be in a named namespace	**5**	Top	Top Suggestions: 1. Avoid using using namespace std, utilize only what you need, eg using std::vector and so on..	The code well written and easy to understand, good job! Suggestions: 1. In shape.cc, distanceTwoPoints is contained in a namespace without apparent reasons 2. In the class Simulation, in order to avoid useless copies and large arrays, use arrays of unique_ptr of robots 3. Avoid passing around the vector containing the robots / particules (even after correcting the missing const), as it is easy to break another time the encapsulation, move the for loops in simulation and let the testing function between two robots, two particules... in the submodules 4. In project.cc, there some useless includes	**9**
363802	**0**	[A1] Pourquoi créer un robot spatial avant de faire la lecture ? Mettre le message de success dans lecture aussi pour le rendu 2. [A2] particule et robot ne doivent pas être inter-dépendants	**3**	Ok	**5**		ok	Le code est très bon, mais quelques erreurs dans l'idée même de l'implémentation sont a corriger pour le rendu 2.	**8**

363823	**2**	Ok	**3**	Ok	**5**		Ok	Excellent rendu, n'hésitez pas à séparer vos conditions multilignes en plusieurs booléens reliés ensuite par des conjonctions logiques	**10**
363889	**2**	Warning - changer l'emplacement du message de succès, ca doit etre dans le modèle et pas dans projet.cc	**3**	OK	**4**	[L1]shape.h, simulation.h, particule.h	Certains modules ne sont pas du tout indentés; Warning - faire attention aux lignes trop longues dans shape.cc, simulation.cc	Bon travail, présentation du code à revoir	**9**
363966	**2**	OK	**3**	OK	**5**	[L1] warning particule.cc: 15,20 [L2] warning robot.h:63 robot.cc:94,119	Attention aux erreurs d'indentation et à la limite de caractère. Quand vos lignes sont trop longues essayer de les aligner par rapport au début des paramètre de la ligne du dessus.	Le code est bien pensé et très lisible. Bravo!	**10**
363979	**2**	Warning: inutile de refaire des includes qui sont déjà dans le header du module robot; Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	ok	**4**	[L1]: simulation.h: 13,19; robot.h: 11,13,23,25,35,42,55,53; particule.h:10,13	Il ne faut pas indenter les mots-clés public, private et protected	Excellente décomposition, code très lisible. Très bien. Faites juste attention de bien respecter les conventions d'indentation pour les prochains rendus	**9**
364036	**2**	Architecture ok	**3**	Ok, no global variable, good use of "const" in the getter	**4**	[L1]simulation.cc 47,59,62,particle.cc 15	Bon style, dommage pour les quelques petites erreurs d'identation	Bon travail beau code. Attention cependant, les switch case ne nécessitent pas forcément de bracket	**9**
364043	**2**	Ok	**3**	Ok	**4**	[L1] particule.cc:34	Plusieurs erreurs d'indentation qui rendent la lecture de votre code moins agréable	Très bon rendu, faites attention à votre indentation	**9**
364060	**2**	ok	**3**	WARNING:using namespace std ne devrait pas apparaitre les .h	**4**	[L1] particule.h: 15,21; robot.h: 14,19,27,32,39,47,55,61	Il ne faut pas indenter les mots-clés public, private et protected	Très bien	**9**
364120	**2**	ok	**3**	ok	**5**	Warning: [L1]: missing indentation simulation.cc 172,		c est biem, continuez comme ça	**10**
364132	**2**	Warning - changer l'emplacement du message de succès, ca doit etre dans le modèle et pas dans projet.cc	**3**	OK	**4**	[L1]shape.h, simulation.h, particule.h	Certains modules ne sont pas du tout indentés; Warning - faire attention aux lignes trop longues dans shape.cc, simulation.cc	Bon travail, présentation du code à revoir	**9**
364150	**2**	WARNING: Include inutile de particule.h et de shape.h dans simulation.h; Warning: epsil_zero hors du namespace shape (peut causer des name collisions)	**3**	ok	**5**	[L1] simulation.cc 20-25,26.	WARNING double indentation: Pas pénalisé car moins de 4 fois	Excellente décomposition. Code très agréable à lire mais respectez l'organisation des fichiers (ex: robot.h les typedef sont avant la définition des classes)	**10**

364154	**2**	WARNING: particule ne devrait pas dépendre de shape, relire la donnée	**3**	ok	**4**	[L1] particule.h: 12; robot.h: 13,19,29,34,39; simulation: 10,20	Il ne faut pas indenter les mots-clés public, private et protected	Bien	**9**
364156	**2**	ok	**3**	ok	**5**		ok	Très bien	**10**
364183	**2**	ok	**3**	WARNING:using namespace std ne devrait pas apparaitre les .h	**3**	[L1] particule.h: 7,11; particule.cc; robot.h: 12,14,18;simulation.h: 11,14 [L2]: robot.cc: 18,40,41,43; robot.h: 30,45,46	Il ne faut pas indenter les mots-clés public, private et protected;il ne peut y avoir 2 indentations (particule.cc), plusieurs dépassements de 87 caractères dans robot.cc et robot.h	Bien mais il y a des erreurs de mise en forme à corriger	**8**