Mini project 1

علی رمضان زاده

اردیبهشت ۱۴۰۳

مقدمه

الگوریتم استفاده شده در این کد DFS است.در این الگوریتم، پیمایش ساختار داده‌هایی نظیر درخت و گراف به‌صورت عمقی انجام می‌شود. روند جستجو با گره ریشه درخت یا گراف شروع می‌شود و برای جستجوی گراف می‌توان از هر گره‌ای کار پیمایش را آغاز کرد.

گزارش کد

در ابتدا کار های اولیه کد را انجام می دهیم اعم از شناخت متغیر ها و سنسور ها دریافت مقدار سنسور ها و ... .با کامنت هر متغیر در کد توضیح داده شده است.

شناخت اطراف خود(۴ جهت)

برای این کار با استفاده از دستور delay ربات را مقداری جلو میبریم تا (به اندازه یک واحد ماز) با استفاده از سنسور ها تشخیص می دهیم دیواری هست یا نه.این کار را برای ۴ جهت انجام داده و می فهمیم ربات چه مسیر هایی را دارد.

کالیبراسیون

به دلیل اینکه از دستور delay استفاده می کنیم ربات از مسیر خارج شده یا جهتش تغییر پیدا کند پس در اینجا از دو حلقه کنترلی کنم می گیریم.

۱- وقتی در حال شناخت ۴ جهت است هربار به دیوار برخورد می کند با استفاده از ۲ سنسور پشت یا جلو سعی می کند عدد دو سنسور را تا حد ممکن به هم نزدیک کند.(کالیبراسیون جهت)

۲- اگر دیوار چپ و روبه رو وجود داشت با سرعتی خود را به گوشه چپ بالا می رساند و با یک delay نسبی خود را به طور تقریبی به مرکز یک واحد ماز می رساند.اثرات حرکت های قبلی را کم می کند و خطای کلی را کاهش می دهد.این الگوریتم را روی بقیه گوشه ها نیز می توان انجام داد ولی با همین یک حلقه ربات به خوبی کالیبر می شود.(کالیبراسیون مکان و جهت)

الگوریتم حرکت در ماز

پس از اینکه از جهت و مکان ربات مطمئن شدیم برای ربات مشخص می کنیم در کدام جهت حرکت کند.برای این کار یک شرط if else می گذاریم که اگر رباتَ چند راه داشت به ترتیب چپ،روبه رو،راست و در آخر به عقب برود.با این کار همه مسیر هارا طی می کند.

به دست آوردن نقشه ماز

در اینجا به چندین مشکل بر می خوریم که مرحله مرحله توضیح می دهیم.

نگهداری جهت های اصلی

ربات ما همواره در حال تغییر جهت است.برای اینکه بتوانیم جهت های اولیه را حفظ کنیم و در نهایت از این جهت ها برای مپ کردن استفاده کنیم یک متغیر زاویه تعریف می کنیم.
این متغیر زاویه چرخیده شده نسبت به جهت اول را می یابد وی با یک گزاره شرطی می توان جهت رفته شده نسبت به حالت اول را پیدا کرد.همین طور برای شناخت دیوار ها نسبت به جهت اول نیز کمک می کند.

به دست آوردن ماتریس نقشه

با توجه به اینکه ابعاد نقشه را داریم(۴*۴) پس یک آرایه ۴ در ۴ تعریف می کنیم. به کمک الگوریتم شناخت ۴ جهت و نگهداری جهت اصلی یک لیست ۴ تایی درست می کنیم و سپس آن راه به یک استرینگ ۴ تایی تبدیل می کنیم برای مثال(۱۱۲۱).به ترتیب از راست نشان دهنده دیوار روبه رو،پشت،راست و چپ است.اگر عدد ۱ بود یعنی دیوار و ۲ بود یعنی دیواری نیست.
حال این عدد ۴ تایی را در درایه اول ماتریسی که ساختیم می ریزیم.البته درایه مورد نظر بستگی به جهت اولیه و مکان ربات دارد که باید به عنوان ورودی به کد داده شود.

ساخت شماتیک مپ در خروجی

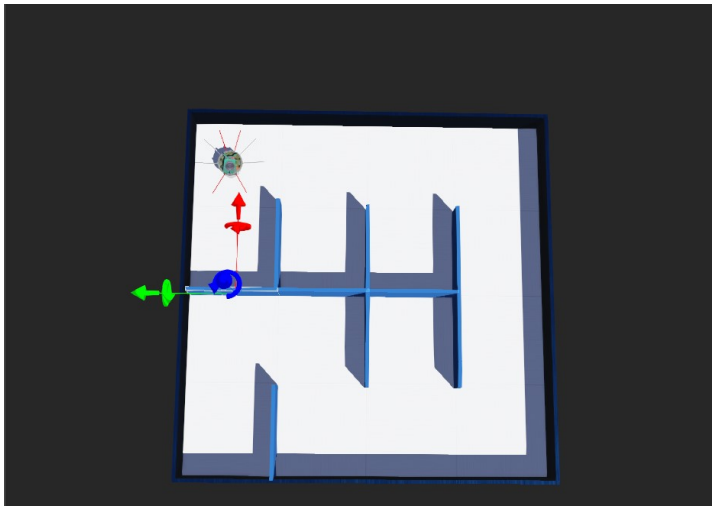
با استفاده از ماتریسی که در بخش قبل به دست آوردیم و گزاره هایی که در آخر کد هست به صورت شماتیک چاپ می کنیم.
یک مثال از مپ،ماتریس خروجی و مپ شماتیک در صفحه بعد آورده شده.

برگشت به نقطه اول

می دانیم یک مپ ما ۱۶ خانه دارد برای حرکت در این ماز ما هربار ۲ بار یک خانه را می گردیم پس برای هر بار تمیل ایت ماز ۳۰ بار حرکت می کنیم.پس کافیست در حلقه اصلی کنترلی یک شمارنده بگزاریم که ۳۰ بار حلقه انجان بگیرد.
پ.ن:اگر نقطه شروع جهت خوبی نداشته باشد یا مکان آن از مرکز کمی فاصله داشته باشد ممکن است بار اول به خوبی نقشه را نساز پس بهتر است شمارنده ما تا ۶۰ برود که نقشه درست خروجی دهد.

اشکالات کد

اگر مکان اولیه که ربات را قرار می دهیم خیلی دور از مرکز باشد احتمال دارد با اختلاف یک خانه بایستد به همین دلیل نقشه خروجی کاملا غلط می شود.
اگر دور در ماز وجود داشته باشد ربات در لوپ می افتد.
می توان بحث کرد که الگوریتم کندی است اگر مجبود باشد ۲ دور بچرخد.



شکل ۱: World مورد استفاده

```
[[1221. 1222. 1222. 1212.]
 [2111. 2111. 2111. 2211.]
 [1221. 1212. 1211. 2211.]
 [2111. 2121. 2122. 2112.]]
* * * * * * * * * * * * * * * *
*                             *
*                             *
*                             *
*     *       *       *       *
*     *       *       *       *
* * * * * * * * * * * *       *
*             *       *       *
*             *       *       *
*             *       *       *
*     *                       *
*     *                       *
* * * * * * * * * * * * * * * *
```

شکل ۲: خروجی کد